University of Twente Newminds make IT happen B.V.

“Improving Newminds’ visual scheduling tool; automation of scheduling”

Master thesis Industrial Engineering and Management

A.C. Koopman BSc.

UNIVERSITY OF TWENTE.

NEWMINDS

"Improving Newminds' visual scheduling tool; automation of scheduling"

Master thesis Industrial Engineering and Management

Author
A.C. (Anniek) Koopman BSc.

University of Twente
School of Management and Governance
Industrial Engineering and Management
Production and Logistics Management

Newminds make IT happen B.V.

Graduation committee

Dr. P.C. Schuur
Lead supervisor
University of Twente

R. Heino / E.Laarman
Company supervisor
Newminds make IT happen B.V.

Dr. ir. J.M.J. Schutten
Second supervisor
University of Twente

B. Hoogland
Second company supervisor
Newminds make IT happen B.V.

Management summary

Newminds provides solutions in the form of web and mobile applications aimed to support delivery of services. One of their current products is the so called “Graphic Planboard” (GP). The GP is a manually controlled scheduling tool. In order to keep pace with competitors Newminds is set to explore automating options in regards to the previously mentioned the scheduling tool. To achieve this end, the reasoning that underlays the scheduling function must be captured and consequently be converted into a functioning scheduling model. This research paper is concerned with identifying the factors incorporated in the existing scheduling function and discovering possible solution methods that can lead to a functional scheduling model. These aspects conclude to form the following research question:

“Whilst enhancing the existing scheduling tool, what solution method to automate the scheduling process is optimal for Newminds, when multiple scheduling influencing factors are taken into consideration?”

The GP is in use for scheduling by customers operating in different sectors. Scheduling refers to the logical sequencing and timing of work to be performed. The implementation of scheduling is subjected to the business environment, strategic company goals, and company preferences. Regardless of the setting, the scheduling function always involves a set of resources and a set of events. Resources are a type of unit which is able to transform a certain type of input into an altered output. The term events represent the jobs or orders that require action in order to be fulfilled. In order to create a schedule, resources and events should be assigned reciprocally. The assignment that ensues is based on the required characteristics in regards to the events and the deliverable characteristics in regards to the resources.

The results of a desk study lead to the conclusion that a constructive heuristic is most suitable for implementation to achieve a rudimentary functional scheduling model. Constructive heuristics recursively construct a set of objects from the smallest possible constituent parts. In every stage, the most optimal choice at that moment is selected by using a priority rule. The main advantage of a constructive heuristic can be found in its universal applicability. Furthermore, the associated solution methodology is considered highly when evaluation criteria as accuracy, speed, simplicity, and flexibility are taken into account.

In order to ensure proper application of the priority rule multiple general characteristics are acknowledged. Several of these characteristics are assigned as a static value; the content of these characteristics cannot change and therefore the value assigned to the characteristic does not change. These priority values must be imposed by the involved company. By doing so the functional scheduling model is adapted to fit the needs of individual customers. The remaining characteristics are assigned a dynamic value; the content of these characteristics can change and its assigned value changes accordingly. There are several possible methods that can be employed in order to construct a schedule based on priority values. In this research four different constructive algorithms are designed as follows:

- Algorithm 1 schedules based on static priority values. It assigns events to the best available resource. New events are always added to the end of the existing schedule.
- Algorithm 2 schedules based on static priority values. It assigns events to the best available resource. New events are inserted into the best position of the existing schedule.
- Algorithm 3 schedules based on dynamic priority values. It assigns events to the best available resource. New events are always added to the end of the existing schedule.
- Algorithm 4 schedules based on dynamic priority values. It assigns events to the best available resource. New events are inserted into the best position of the existing schedule.

Algorithm 4 has emerged as the best option from a computational experiment. This procedure outperforms the other algorithms in several aspects, specifically the number of events scheduled, efficiency of spent time and total priority value scheduling. The amount of CPU time that is associated with algorithm 4 is slightly larger than the time consumed by its contestants, yet the attached benefits to this algorithm are considered to outweigh this factor. Based on these results, Newminds is recommended to develop a functional scheduling model that creates a schedule that adds the most valuable combination to the best position in the existing schedule by using dynamic priority values. Additionally, this solution methodology has the advantage of relatively easy expansion if desired. This can be realized by adding more side constraints or including more scheduling characteristics.

Preface

Hereby I present to you the final report for the graduation of the master program Industrial Engineering and Management at the University of Twente. The research is conducted on behalf of Newminds make IT happen B.V., location in Hengelo.

The past year and a half I have been working on this report. Luckily I did not have to do it all by my own, and therefore I would like to take the opportunity to thank some people. First of all, I thank Newminds for giving me the opportunity to perform this research at their company. Especially I thank René, Erik, and Björn for their support and their honest feedback.

Next to Newminds, I owe gratitude to my supervisors of the University of Twente. Peter Schuur helped me from the beginning of my search for a graduation project and all the way through performing my graduation project, and for that I thank you. I'm also grateful to Marco Schutten. Although he was not involved at the start of the project, he really helped me by giving feedback.

Last but not least I owe gratitude to my friends and family. The process of conducting research was not always easy, but no matter what, they supported me and helped me through. Especially I thank Pieter Koopman, without his help I would not have been able to construct the environment for testing my algorithms.

Anniek Koopman

July 2016

Abbreviations and definitions

AI	-	Artificial intelligence
AO	-	Alternate operation
EDD	-	Earliest due date
ERP	-	Enterprise resource planning
GP	-	Graphic planboard
ISIS	-	Intelligent scheduling and information system
IT	-	Information technology
LPT	-	Longest processing time
MmTSP	-	Multi depot multiple travelling salesman problem
MSI	-	Minimum stress insertion
mTSP	-	Multiple travelling salesman problem
Navision	-	Microsoft dynamics Navision
NN	-	Nearest neighbour
OPIS	-	Opportunistic intelligent scheduler
RCSP	-	Resource constrained scheduling problem
SGS	-	Schedule generation scheme
SLA	-	Service level agreement
TSP	-	Travelling salesman problem
TTET	-	Composite travel time expiration time
TTS	-	Time transcending schedule
VRP	-	Vehicle routing problem

Table of Contents

Chapter 1 Introduction

This research, commissioned by Newminds make IT happen B.V. (Hengelo, the Netherlands), is performed in the framework of completing the Master program Industrial Engineering and Management, specialized in production and logistics management, at the University of Twente. As information technology (IT) is the core element within this organization, and as IT is developing fast, it is important for Newminds to develop new and improve existing products. One of the developments they want to achieve is the automation of the current scheduling tool. This graduation project comprises the development of the functional model that is required in order to be able to automate the scheduling process.

This chapter provides an introduction to the company and the performed research. In Section 1.1 the reader gets an introduction to the company and the main product in this research. The remaining sections of this chapter describe the research outline: Section 1.2 gives the research objective, followed by the problem description in Section 1.3.

1.1. Newminds

Newminds is a small (i.e., approximate 20 employees), IT-based company located in Hengelo, Overijssel. They develop and implement enterprise resource planning (ERP) systems, primarily based on Microsoft Dynamics Navision (Navision). Navision is global ERP-software that supports companies managing their finance, supply chain, and operations. The main part of Newminds' customers operates in the field service, food or production sector.

In addition, they develop and implement web- and mobile applications, including the connection between those applications and the ERP-system. In order to deliver an integral solution to the customer, forces are joined with partner Newminds Systems. Whereas Newminds focuses on the development and implementation of ERP-systems and applications in relation to software, Newminds Systems focuses on the hardware side supporting the software.

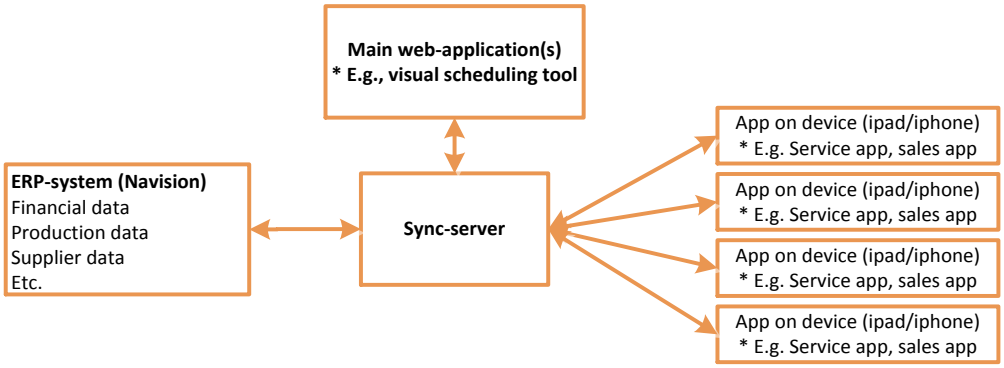


Figure 1.1 Newminds' data structure

The products designed by Newminds consist of multiple elements; there is the ERP-system itself, the synchronization server (sync-server) and several applications (see Figure 1.1, the exchange of information is indicated by arrows). Together these elements provide an infrastructure that delivers an integral solution to the customer and so these elements are sold as one product. Although all elements

are designed and implemented by business units of Newminds, the products are brought to the market under the name Spatio. However, in this research all products are referred to as Newminds' products.

One of the applications created by Newminds is their visual scheduling tool called "Graphic Planboard" (GP). This tool supports planners to keep track of the scheduling process, which can be in any sector. In this research the focus is on this application. Subsection 1.1.1 introduces the design and functions of GP. Subsection 1.1.2 gives insight into the current practice of the tool. Subsection 1.1.3 gives an introduction to (desired) improvements.

1.1.1. Newminds' supportive scheduling tool

Over the years different players developed multiple scheduling tools in order to support companies' scheduling processes. In order to compete with these players, Newminds developed their GP. Figure 1.2 shows how GP is displayed to its user and highlights the most important elements.

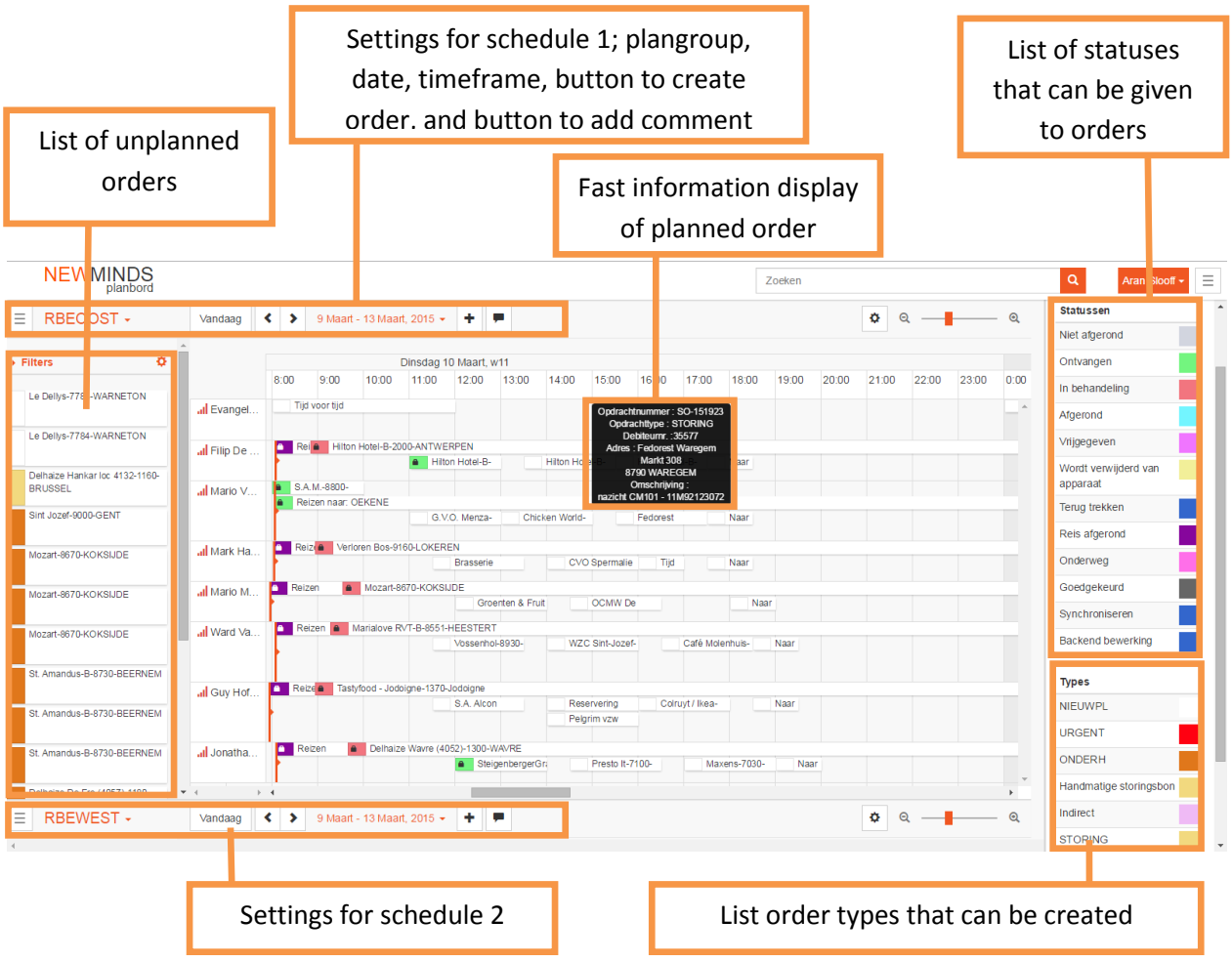


Figure 1.2 Newminds' GP with comments on functions

GP provides planners insight into the current schedule, and enables them to (manually) adapt the schedule to preferences and real-time requests. The information displayed in GP is derived from the customers' ERP-system. Most of Newminds' customers use Navision as their back-office system as

Newminds offers seamless integration between those elements. However, in combination with other ERP-systems the connection between such ERP-systems and GP is modified in order to function.

When using GP, multiple information elements are directly shown to the user. The sides show information panels (e.g. list of orders, list of statuses, setting bar), while the centre displays the schedule. A planner initializes the settings for the schedule. Those settings relate for example to the timeframe for which the schedule is given (e.g., a day, 2 days, 3 days, or 5 days), the start date of the timeframe, the display size of the schedule, and the plangroup for which the schedule is created. A plangroup refers to a specific subset of resources that are distinguished by the company. Examples of plangroups are North, East, South, and West. Each plangroup has its own resources, and one generates schedules per plangroup as they are independent of each other. All these preferences are gathered in a first setting bar. A second setting bar is available and enables a planner to display two schedules at the same time.

Within the ERP-system orders are created and assigned a type and status. The status of an order is variable (i.e., it changes over time), the type of an order is fixed. Based on expected travel and working times a schedule is created. Orders performed in the past are still shown in the system. However, GP then contains real data concerning travel and working times instead of expected travel and working times.

By placing the cursor on an order, a fast information display opens (see Figure 1.2). This display gives a summary of the order information. When selecting an order, the system shows more detailed information of the order, and the user can add notes. Orders that are not yet assigned to a resource are displayed on the left. By using a drag-and-drop function orders can be scheduled at any resource at any time.

GP is connected to an ERP-system which provides the scheduling data. The offered connection between the ERP-system and the scheduling system is one of the strongest elements of GP. Planners usually use an administrative system and a scheduling system in a parallel way. GP automatically connects the data related to orders on both systems. This integration of both systems saves the planner effort and time, and limits the risk of failure in copying information.

Another advantage of GP is that changes, or any other disruptions in the process, are directly visible and can immediately be taken into account. As the system is accessible from any computer device with an internet connection, the system is easily accessible. The accessibility comforts users to keep track of the system state easily, and to perform required actions.

In addition to the seamless connection to the ERP-system, GP seamlessly integrates with the Newminds' service app. This app enables users to gain insight in the real situation, and to update the schedule according to the live information that is received. The app tracks the status of current activities and guides service employees to their next job. Within the app service employees record times for travelling and working and define required or used materials. Based on this information, a planner updates the schedule and sends the new information to the service employees' mobile devices.

Another related app is Newminds' sales app. Within this app salesmen state agreements related to the deals that are made on location. The required information for the visit is accessible within the app and the added information is directly visible in ERP. The visits are scheduled within GP.

Newminds is responsible for the functioning of the current products, and so they perform maintenance and debugging actions. Besides this, they elaborate their current existing and new products in order to keep up with competitors or create innovation.

1.1.2. Current practice of the supportive scheduling tool

While discussing the function of the supportive scheduling tool, both scheduling and planning are often used to describe the same type of activity. According to Oberlender (2000) scheduling refers to the logical sequencing and timing of the work to be performed, as planning refers to the formulation of a course of action (i.e., explicit operational plan) to guide a project to completion. In general, planning is used for strategic (i.e., long term) and tactical (i.e., medium term) control of the process, while scheduling focuses on the operational (i.e., short term) control. In addition, scheduling and planning refer to an action; a schedule or plan is the result of the action. In this research the primary focus is on scheduling, as the focus is on short-term processing instead of projects or long-term actions. This implies that attention must be paid to the sequencing and timing of work.

GP can be applied in multiple sectors, whether it is about scheduling external operating service employees in a service organization, production orders within a manufacturing company, sales visits in relation to contract management, or trucks within a transport company. Most current users of GP operate in the service or sales environment.

The scheduling process differs per company as it is based on the specific situation of the company, company preferences, and the strategic goals of the company. The scheduling process results in some schedule. In this research a schedule is defined as "a plan for performing work or achieving an objective, specifying the sequence and allotted time for each part" (Farlax Inc.). Only if there are no orders to be (re)scheduled, the scheduling process stops. One should consider whether it is needed to have a scheduling function, as the benefits of having such a system might be less than its cost. However, in many businesses a scheduling function is required in order to structure the process.

Although the company characteristics might differ, scheduling itself is always performed. This implies that the core function of scheduling is the same and that the scheduling process is (partly) independent of the environment. In this research the primary focus is on three business environments: *production, service* and *sales*. The decision to focus on these business environments, and to not consider for example trucker logistics or other scheduling processes, is due to the business environments of Newminds' current customers.

Based on the main characteristics of the scheduling function in the different environments, a generic scheduling entity is created in Chapter 2. This generic scheduling entity is the basis problem formulation for which a solution must be found. By structuring the search for a solution like this, the solution is widely applicable as desired by Newminds.

Scheduling in a production environment

Within the production environment, the focus is on scheduling production orders on producing units (i.e., machines). For the completion of an order, often multiple producing units are involved as multiple production stages must be performed. The system should determine in what sequence a producing unit has to process production orders. This is based on at least the following factors: expected duration of a task, release and due date of an order and the required product routing over machines.

Pinedo (2005) describes the scheduling function in production organizations as: “in a production and manufacturing model, resources are usually referred to as machines and tasks that have to be done on a machine are usually referred to as jobs”. Although the focus in the production environment is on machines, also human resources are important. Human resources are for example required to operate the machinery.

According to Pinedo jobs are released to the system, each associated with a release and due date and a specific sequence to follow over multiple machines. This specific sequence which must be followed is referred to as *routing.* Routing in this case is based on the activities. The processing of jobs may be delayed or interrupted in case certain machines are busy, if machines break down or when high-priority orders are released and mess up the processing sequence of jobs. Scheduling involves the challenges of sequencing jobs such that the overall performance is at his best.

Planning and scheduling in a service environment

The focus in a service environment is on scheduling service employees, who process orders by visiting customers. The system should determine what service employee to assign to what customer in order to visit all customers once. At least the following factors must be considered while assigning customers to service employees: moment in time, expected duration of task in relation to time frames, service employee skills, and location.

Pinedo (2005) identified additional characteristics, compared to the production environment, that occur while scheduling in the services environment. These characteristics are: generally no goods are kept as inventory, yield management is more important as denying a customer is undesirable, and the number of resources may vary over time. Activities in services are associated with release and due dates, in addition a set of activities is associated with *sequencing*. Based on the cluster of activities assigned to a specific resource, the order in which they are dealt with is referred to as sequencing. *Clustering* describes the assignment of activities to a specific resource. This can be done based on different criteria, and the size of a cluster can fluctuate over time. Besides, activities can shift cluster as the overall set of activities changes.

Planning and scheduling in a sales environment

The focus in the sales environment is on scheduling sales employees who visit customers in order to deliver a product or service. The system should determine what customer to assign to which sales employee based on at least the following factors: moment in time, expected duration of task in relation to time frames, sales employee skills, and location.

Scheduling in the sales environment is very similar to scheduling in the service environment. However, the occurrence of related administrative activities, referred to as to-dos, is more common in the sales environment. These activities must most often be performed by the same resource that performed the main activity. The main activity is performed at the customer's location, the administrative activities are performed at another location (e.g., at home or at the company office). The occurrence of to-dos is a main difference compared to scheduling in a service environment, but as administrative activities might also occur in the service environment it is not very distinctive. In general there is no time scheduled to perform specific to-dos, they only have a creation date and a due date. The schedule might contain time-windows that are blocked for the sales employee in order to process to-dos, but the sequencing and selection of to-dos to perform is up to the sales employee.

In this environment sequencing is also relevant. However, sequencing is often solved on individual basis, with a specific set of customers that is assigned to a specific sales employee. Once a customer is assigned to a sales employee, most often all other visits to the same customer are performed by the same sales employee. The way activities are clustered, and so assigned to employees, is different compared to the service environment.

1.1.3. Development of the supportive scheduling tool

The current version of GP requires manual actions of a planner to (re)schedule. While scheduling, a planner takes into account multiple factors in order to create a feasible schedule. At the moment of writing, Newminds is expanding the system options and adding some features, in order to create better insight for the planner in the feasibility of scheduling options. One of the features that is added relates to the visual display of infeasible scheduling options. For example, if employee X is not working on Mondays, Monday will turn grey for this employee. Also, if an order requires certain skills, the timeline of all resources that do not have those particular skills turn grey if that particular order is selected.

In general, the decision-making process that is fundamental in planning or scheduling often relies on mathematical techniques and heuristic methods to allocate limited resources to the activity that has to be done. The allocation has to be done in such a way that the company optimizes its objectives and achieves its goals (Pinedo M. , 2005). Newminds explores possibilities in automating the scheduling system but therefore the reasoning behind the functioning of the scheduling function must be captured. Once the reasoning is captured, a function could be programmed that enables one to create a good schedule by just pressing a button.

Based on this information, three development stages for GP can be distinguished. As a basis there is the current state of GP. Current developments focus on creating a supportive scheduling tool (i.e., stage 2). After completing this stage, Newminds wants to automate the scheduling process (i.e., stage 3). Figure 1.3 shows the system requirements for each of these stages, as indicated by Newminds.

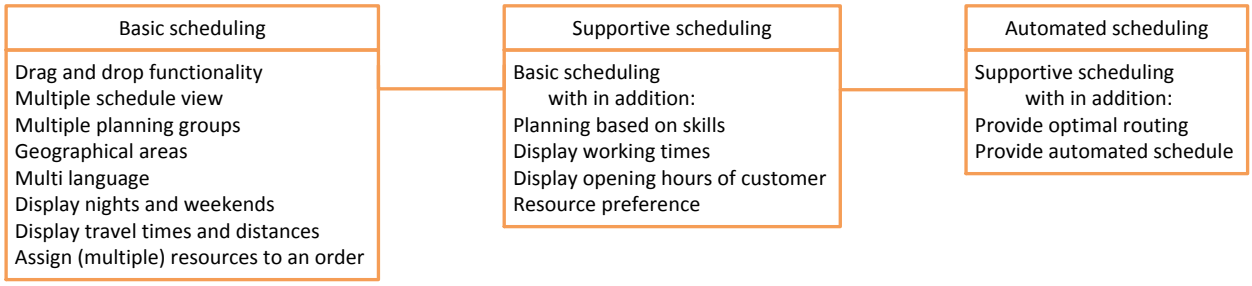


Figure 1.3 Development stages of the visual scheduling tool

In order to design an automated, widely applicable scheduling system, the fundamental decision-making process for scheduling in organizations must be mapped. The scheduling function may differ between organizations, for example based on the product or service they deliver, but one may also expect similarities. The similar decision factors should serve as basic function for automated scheduling.

The automated scheduling system provides current and future customers with a tool that makes scheduling easier, more accurate, and less scheduling- resources consuming. The potential of the system is large, and if created successfully, Newminds could really benefit from this as it creates a superior market position compared to their competitors. The factors that influence the decision-making process in scheduling are therefore of great importance. These factors, how they influence the scheduling process, and how they can best be designed in an automated scheduling system must therefore be identified.

For the design of the automated scheduling tool there are some constraints. The first and most important constraint is that the automated scheduling tool can be implemented within the current information technology infrastructure. This implies that no current relations between software applications, or servers, are changed, and that no additional servers are required. The only addition that is allowed is the processing-element that deals with the scheduling process (i.e., scheduling engine). Figure 1.4 shows the (desired) position of this new element.

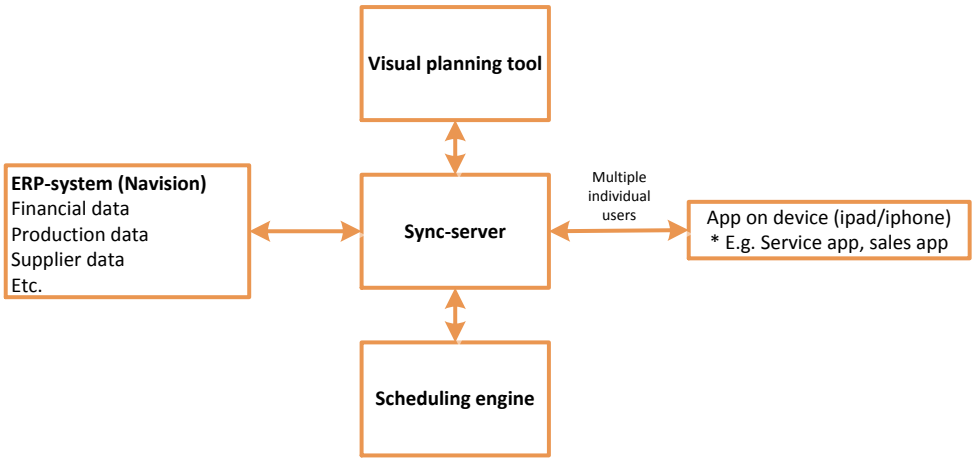


Figure 1.4 Data structure including the desired position of scheduling engine

Second, the automated generated schedule should represent the current way of scheduling. Third, the tool should generate a feasible schedule within reasonable time. For this matter the possibilities and

limitations related to GP's data infrastructure, and GP's data processes, are considered. As the generated schedule must be feasible at any time, the use of real-life information is required, but offline gathered data can be used to build the schedule initially. The amount of time that is reasonable for generating a schedule depends on the used strategy: a schedule involved with little uncertainty, and so less expected (re)scheduling actions is allowed to take more time compared to a schedule that requires more rescheduling actions. This because the quality of the schedule within a stable environment is not likely to change, and therefore it must be good the first time. When rescheduling is required, the schedule created at first is likely to be changed and changes should be made instantaneously. Therefore (re)scheduling is not allowed to take long.

1.2. Research objective

This research maps the different factors that influence scheduling, and consider their representation within an automated scheduling system. While considering those factors, the interaction between those factors and their (individual) influence on the solution are taken into account. Based on those factors a model is built that results in a good and feasible schedule.

Basically there are three research elements. First the input characteristics are identified. These input characteristics relate to the set of orders, and the set of resources that are involved in the scheduling. The second element relates to the way those characteristics are dealt with in order to make scheduling decisions. Once it is known how to make scheduling decisions, those decisions are mapped in a model. The third element translates the decisional model into a useful output (i.e., a schedule). Figure 1.5 gives a schematic overview of the research elements.

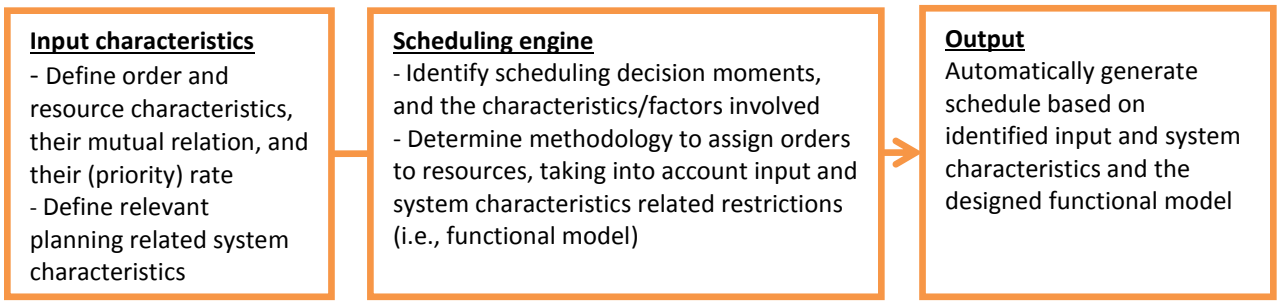


Figure 1.5 Schematic overview of research elements

The overall objective of this research is to come up with a functional model, the "scheduling engine", which takes into account multiple factors and can provide GP users with an automatically generated schedule based on the identified factors.

As it is Newminds' desire to apply the automated scheduling tool in multiple business environments, the similarities and differences between these environments must be considered at first. The common factors determine the scope of the research and the functional model.

In case of service related business, the focus is on scheduling service employees who visit customers to deliver the product or service. The delivery of a product or service is defined as a service order. The system should determine what customer is assigned to which service employee. In case of production related business, the focus is on scheduling production orders to producing units (i.e., machines). The

system should determine the sequence of producing units. In case of sales scheduling, the focus is on scheduling sales employees visiting customers. The system should determine which customer is assigned to which sales employee in order to visit all customers once.

These environments might look different, but each case can be simplified to (at least) a set of resources and a set of orders needed to be processed. However, the service and sales environments primarily focus on the assignment of orders to resources, while within the production environment the primarily focus is on sequencing orders on resources. In Chapter 2 the input values, related to resources and activities, are generalised. This generalisation is used to demarcate the characteristics of resources and activities that are dealt with in this research.

1.3. Problem description

To automate the scheduling process, knowledge should be gathered about the scheduling process, the reasoning behind decisions in this process, and the factors that influence these decisions. Since different factors can conflict with each other, optimizing a schedule is a difficult task. Based on the current knowledge on the research problem, the following central research question can be derived:

"Whilst enhancing the existing scheduling tool, what solution method to automate the scheduling process is optimal for Newminds, when multiple scheduling influencing factors are taken into consideration?"

This central research question cannot be answered immediately, and therefore the research is divided into several sections. Each section contributes in answering the central question. The different sections are distinguished based on the initial project planning (see Appendix A). Below, for each chapter the main theme is outlined, and the deliverables are identified. These deliverables should ensure progress, as at the end of a chapter it is checked whether the deliverable is met, and so if the chapter covers the elements that are required.

Chapter 1 gives an introduction to Newminds, their scheduling tool, and the use and functions of the existing scheduling tool. In addition, the requirements and limitations for the design of the automated scheduling function, and the general context of the research are identified.

Deliverable 1a: Identification of core environment characteristics for scheduling

Deliverable 1b: List of design requirements for automated scheduling

Chapter 2 focuses on the identification of input characteristics. For this matter the core environment characteristics for scheduling are identified. At the end of this chapter a demarcation of input characteristics is made.

Deliverable 2: List of general factors that influence the scheduling function

Chapter 3 contains the results of a desk study. Information models and solution methodologies used for solving scheduling problems similar to the research problems are given. For the models and methodologies found an examination is done on (what part of) those models are usable in order to solve the research problem. At the end of this chapter the directives for a (customized) solution must be identified.

Deliverable 3a: List of algorithms that can be applied given the identified factors and the system design requirements

Deliverable 3b: (Part of) suggested method, or combination of methods that is most favourable to apply in automated scheduling

In Chapter 4 a basic solution design is proposed, which is based on the directives that follow from Chapter 3. For that matter the solution methodology design and the design of required parameters is discussed.

Deliverable 4a: Identification of parameters and their design

Deliverable 4b: Identification of restrictions and their design

Chapter 5 is dedicated to the design of the solution methodology. In previous chapters the design theoretical design of the solution is stated. Within this chapter multiple practical solution approaches are proposed and tested.

Deliverable 5a: Outline (step-by-step) of basic solution algorithms

Deliverable 5b: Evaluation of the performance of basic solution algorithms

Chapter 6 provides guidelines for the customization and implementation of the automated scheduling function. Also in Chapter 6, recommendations for the further development of the automated scheduling function are stated.

Chapter 7 summarizes all relevant actions, demarcations, discoveries and choices are presented.

Chapter 2 Generalisation of input values

It is Newminds' desire to create an automated scheduling tool that can be applied in the three business environments as identified before. Based on their short description in Chapter 1 it becomes clear that scheduling always involves at least some resources and some actions.

The characteristics of both resources and actions influence the scheduling function; actions of a certain type, or an action from a certain customer, might be more important to the company than others. Besides, based on same characteristics a resource might be unable to perform a certain action. An example of such a characteristics are skills. The different characteristics of resource and activities imply that the set of resources, as well as the set of activities, are not homogenous.

Section 2.1 acknowledges the general and distinctive characteristics defining a resource or event. Within Section 2.2 sources that interrupt the scheduling process are given. Then in Section2.3 a demarcation is made of what characteristics and factors are taken into account in this research. The focus is on the identification of characteristics that are present in multiple environments. In this research these common characteristics form the basis for the scheduling process. In a later stage the basic model can be expanded by adding more complex variables, more constraints, and more environmental specific characteristics. These details are neglected at this point as the goal is to provide a solution that is widely applicable.

2.1. Standard situation description

Pinedo (2005) addresses the scheduling function in both manufacturing and services. Several elements are used as guidelines for the analysis of the scheduling problem in order to create a standard model. The list of characteristics below is limited and can further be expanded by adding lots of details. However, in deliberation with experts most common and most important characteristics are identified. These experts are all well experienced in working and scheduling in one or more of the discerned business environments.

Pinedo (2005) identifies multiple types of manufacturing scheduling models. For example, there is project planning, single/parallel machine scheduling (job shop models), automated material handling models, lot scheduling models and supply chain planning models. In relation to Newminds' customers, the category of job shop models is most suitable.

For service scheduling also multiple models are distinguished: reservation systems and timetabling, tournament and broadcast television scheduling, transportation scheduling, and workforce scheduling. In relation to Newminds' customers, the category of transportation scheduling is most suitable. This due to the fact that traveling is involved, and the feasibility of (round-)trips must be considered.

As there are differences in the formulation, in the objective, and in some constraints and assumptions, Pinedo states that the algorithms used for scheduling services tend to be completely different from those used in production. However, there are lots of similarities and therefore a common basis is available. The amount of overlap between manufacturing and service models depends on the system characteristics on both sides.

While scheduling in a production environment, one generally speaks about jobs and machines. While scheduling in a service or sales environment, one speaks of visits instead of jobs, and of service/sales employees instead of machines. Features of jobs and machines can often be translated to the context of the service/sales environment. For example, production jobs having a release date, a due date and a processing time are equal to the pop-up date of required service, the (service time agreement) due date and processing time of service jobs, and the skills of a service employee can be compared to the functions of a machine. A generalization of characteristics, given per environment, can be found in Appendix B.

Each environment has its own type of deliverable (e.g., production orders, service calls, and jobs), however each deliverable is related to some product or service which requires some action. From now on each deliverable is described as an *event*. In a broad sense there are multiple types of resources, for example consumable resources, processing resources, etc. The type of resources that is primarily concerned in scheduling are the processing resources. These resources are in this research referred to as resources.

2.1.1. Resource characteristics

Resources are some type of unit that are able to transform some type of input into a different output. In general two kinds of resources can be distinguished: material resources and human resources. Material resources can be machines as well as tools. Production materials are not classified as processing resources, they are classified as consumable resources. To distinct different types of resources, one can decide to assign each resource to a *resource group*. A resource group distinguishes different pools of resources based on resource characteristics such as type (i.e., material or human), when/where they are used, or the resource his function.

At Newminds' customers, the following most common resource characteristics are observed:

1) *Availability: relates to the accessibility of the resources, related to time issues*

   The availability of a resource is influenced by multiple aspects, but at the end it comes down to a yes-or-no question. Examples of aspects that influence the availability of a resource are the operational hours and (ir)regular downtime. In general a resource has a fixed number of operational hours. A machine is for example functioning every day from 8AM to 6PM. However, the general availability of a machine is influenced by planned downtime such as maintenance, and irregular downtime such as machine failure. Regular downtime can easily be scheduled, for example every Monday from 8AM to 10AM maintenance is done. This is in contrast with irregular downtime which comes unexpectedly, and can therefore not be scheduled. The operational hours of a resource might be flexible as resources might be able to work in overtime. If this is the case, costs for additional operational hours should be considered.

   Other types of resources, such as a service or sales employees, have the same factors that influence the availability but often they are referred to in a different way. The operational hours of a machine are similar to the working hours of an employee, planned downtime is similar to days-off and holidays, and irregular downtime is similar to illness.

In addition to the availability there are two sub-characteristics observed at Newminds' customers:

a) A company often has resources that are owned by them, and so these resources are available to them. If additional resources are required, activities can be *outsourced* to external resources. Outsourcing is often expensive. To distinct owned resources from optional external resources, Newminds customers assign a resource type to them. The resource type indicates whether a resource is owned by the company or not. This characteristic is not always applicable or required.
b) In addition *plan groups* are observed at Newminds' customers. The plan group indicates the allocation of a resource to one of the distinguished areas. For example, the total operational area is divided in four areas; North, East, South and West. A resource belongs to one of these areas, and scheduling is done based on these areas. Now each area is a different plan group.

2) Capacity: relates to the ability (i.e. maximum) to receive, hold, or contain something

The capacity of a resource is related to the capability of a resource to produce output per period of time. Resources of the same type and with the same configurations might have a different capacity. The capacity of a resource influences the duration of an event. For example, service employee 1 performs event type A in 30 minutes while service employee 2 requires 45 minutes to perform the same event. The hourly capacity of service employee 1 equals 2 events, the capacity of service employee 2 equals 1.33 event.

It might occur that multiple resources are, or can be, required by an event. For example, a production batch of 300 items needs to be packed. Employee 1 can pack 50 products an hour, while employees 2 and 3 can both pack 40 products an hour. If only employee 1 is assigned to pack products, it takes 6 hours to pack all products. If only employee 2 or 3 is assigned to pack products, it takes 7.5 hours to pack all products. If employee 1 and one of the other employees are assigned to pack products, it takes approximately 3 hours and 20 minutes to pack all products. The time needed to process a certain event therefore depends on the capacity of the resources assigned to it.

3) Utilization: relates to the (effective) use of resources in relation to their capabilities

A resource might be available from 8AM until 6PM, but only might be used 90% of the time. The 10% left is reserved to for example cope with irregularities that occur during the processing of planned events. The utilization of resources might be bounded by legal restrictions, such as workforce laws. The utilization rate is therefore often a decision based on strategic and tactical inputs. However, these inputs are decisive for the scheduling process and must therefore be considered.

4) Skills: relates to the ability to process/perform due to the presence of techniques/knowledge attained by study or practice (i.e., qualifications)

For a machine, skills are related to the properties or configurations of that specific machine. For an employee, skills are related to the ownership of certificates. In general skills indicate whether a resource is able to perform a certain task. Some might see the level of experience of an employee as

a skill. However, experience relates to the amount of tasks that the employee can performed within a given timeframe or the quality of performing a particular task. It does not relate to the basic ability of performing a certain task. Therefore experience is not primarily considered as a skill.

5) Location: relates to the (geographic) position of the resources.

The location of resources is important in the service and sales environment. In these environments, the location of a resource is flexible. Human resources are often assigned to a home-location, as well as a temporary location. The home-location indicates the start- and end location of a resource. The temporary location of a resource indicates the current location of a resource, for example a customer location where the employee is currently present.

2.1.2. Event characteristics

As mentioned before, the required characteristics of a resource, as demanded by an event, should match with the available resource characteristics. An event can therefore be characterised by what is required. An event requires at least some time (i.e., availability of resources, and the allowed utilization of them), some skills, and some space (i.e., a location). In addition to those characteristics, there are some event specific characteristics.

1) Event type: indicates the type of action that has to be performed

The event type can be used to indicate the priority of an event. Examples of event types are installation, maintenance, or malfunctioning. If there is little time left to perform an event, malfunctioning events are more likely to be started compared to maintenance events. This because in the first case the product or service is not functioning and recovery is required, while the product or service for maintenance events are still functioning. In general the functioning of a product or service is critical for the business and therefore most important.

The event type might also influence the availability of resources. For example, if regular working hours are from 8AM to 5PM for all event types, rules can be set that only allows the processing of certain event types outside this time window. Malfunctioning events for example are likely to be performed all day long, while maintenance events are not likely to be scheduled outsider regular working hours.

2) Event status: indicates at which development phase an event is

Examples of event statuses are released, received, in progress, and completed. An event cannot be processed by any resource until it has a certain status: an event that is not released cannot be performed, and so it can never reach the in progress status. The list of possible statuses, and how they limit scheduling, is company specific. The status of an event is important considering the feasibility of an event to be scheduled at a certain time.

The event is often divided into three phases: preparation of the event, a main activity and completion of the event.

The *preparation* of the event must be performed before the real event can start. It relates to fixed duration preparation activities as well as to variable duration preparation activities. Fixed duration preparation activities are for example reading the project documents or filling the machine. Variable duration preparation activities are those activities that depend on the sequence in which events are scheduled, an example of this are travel times. Related to the preparation of an event are preceding events. Preceding events are those events that must be performed before the other event can be started. The difference between preparation activities and precedence events is basically the relation to the main activity. Preparation activities are directly followed by the main activity, preceding events involve activities that are not directly followed by the main activity and therefore they can be seen as a standalone event.

The main activity of an event relates to the delivery of the requested product or service. Examples of main activities are the installation of a machine, bringing a customer visit or producing a product.

Completion activities are required to really complete the event. Activities that are related to the completion of an event are for example writing a visit report, cleaning a machine, or sending the invoice. The completion activities are always a result of the performed main activity.

A performed event can be a preceding event for another event; this implies that the event must be performed before the other can start. Therefore an event that is currently being processed can only have successors, as all its predecessors are already performed. Successors cannot be scheduled before all its predecessors are finished. Predecessors and successors set sequencing restrictions for scheduling a single event. Therefore the relation between events must be considered while scheduling. It is assumed that all phases of performing one event must be performed by the same resource, predecessor or successor activities are allowed to be performed by another resource.

3) Processing stage: related to the multiplicity of processing stages

A deliverable might require multiple events before it is finished. The different events form different stages of the process. Processing stages are often applied in the production environment, where a product needs to be processed by multiple machines (in a given sequence) before it is finished. When dealing with multiple stages, the required *routing* over different types of resources is often predetermined. However, which particular resource of a certain type to use (i.e., assignment of an event to a particular resource) is not predetermined. Within the service or sales environment, the assignment of events to resources is not often predetermined however in person there can be preferences. So although routing is fixed, the assignment of resources to events is important in scheduling.

4) Release and due dates: related to the occurrence and deadline (i.e. availability) of an event

The release date of an event represents the moment in time from which an event can be processed. The due date of an event represents the moment in time by which an event has to be completed. An event can be created today, with a release date of next Monday and a due date of next Friday. This implies that the event has to be processed somewhere next week.

5) Location: indicates where the event should be performed

In production the location of the event equals the location of the assigned machine. In the service and sales environment, the action often has to be performed at the customer location.

6) Allowance of pre-emption / allowance of in-process inventory

These characteristics determine whether an event can be interrupted while being processed and whether time between processing stages is allowed.

Matching resources and events

In order to schedule, one should assign events to resources, or the other way around. This must be done based on the required characteristics on one side, which match the delivered characteristics on the other side. Figure 2.1 gives an example, where event *i* has to be assigned to a resource.

The example event has two preparation activities and one completion activity. One of the preparation activities is identified as being variable. This implies that the duration of the preparation event depends on the sequence in which other events are scheduled on the same resource before this particular event is performed. This relates to for example travel times of employees.

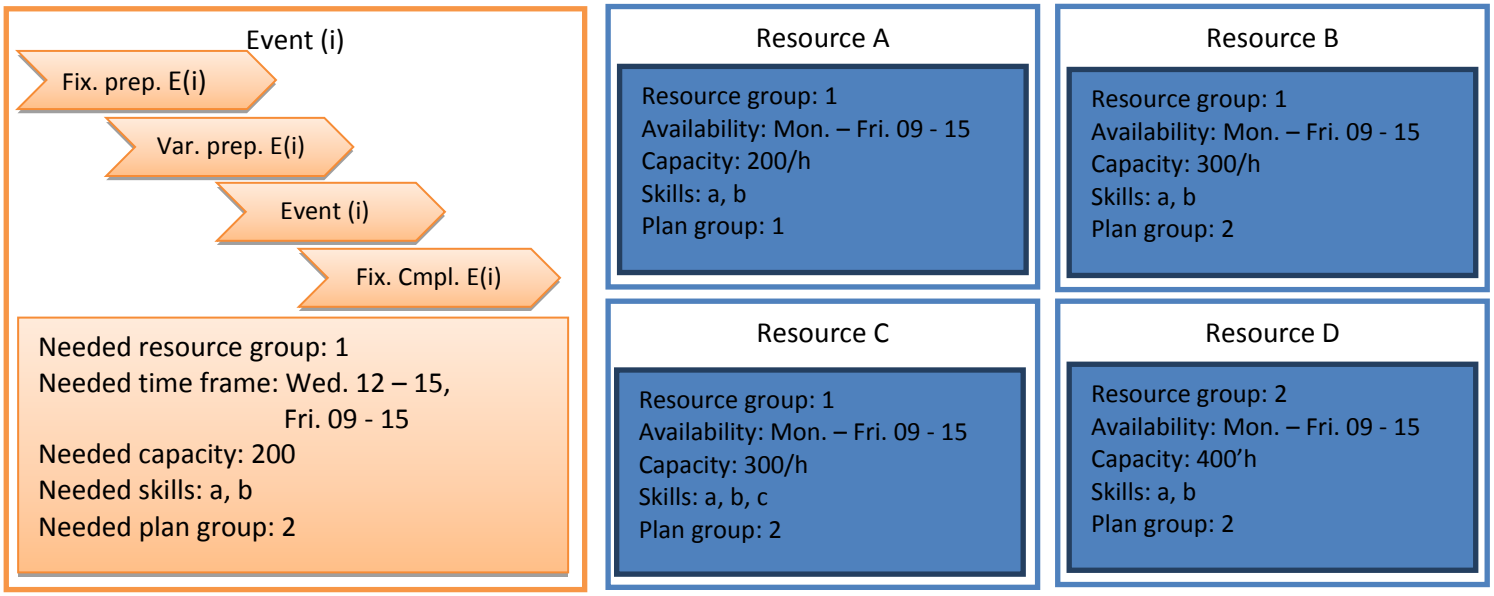


Figure 2.1 Event and resource overlapping characteristics

Based on the characteristics that should overlap between an event and a resource, resource B and resource C are able to perform event *i*. Resource A is not able to be perform the event as it belongs to another plan group, resource D is not able to perform the event as is belongs to another resource group. Whether to select resource B or resource C depends on the other characteristics identified, such as utilization of the resources and (sequence dependent) preparation times. The occurrence of other events therefore influences this decision.

A resource can be assigned multiple events. Figure 2.2 represents a schedule for a resource, independent of the environment. Here an undefined number of events (i.e. "x") is assigned to this resource in a given timeframe. The amount of events assigned to the resource depends on the characteristics of both resource and event, and the way those characteristics overlap. For the graphical representation it is assumed that the events follow one after another without waiting times in between, in reality this is not necessarily the case. Those waiting times cannot always be avoided, but if possible they should be avoided. Therefore they must be considered while scheduling.

Figure 2.2 General plan/schedule representation

Matching the needed resources to events becomes more complex as multiple resources are required, or as event configurations strongly limit the possibilities. Also, if there are more resources that match with the event than required by the event, a selection method must be applied to select a resource. If these decisions are required, they must be included in the scheduling function.

2.2. Sources of uncertainty

Complexity in scheduling is caused by uncertainties that are related to the input (i.e., resources and events) and the environment of the scheduling function. Uncertainty is defined as the perceived inability to predict something accurately, and is characterised by the inability to develop a probabilistic estimate, or a list of all possible outcomes, related to a decision or an event (Miliken, 1987). Uncertainties form a risk to the effectuation of the schedule, and so sources of uncertainties (e.g., risks) must be managed in order to create a more-or-less stable schedule.

Several authors, such as Vasile and Vladut-Severian (2013) and Manuj and Mentzer (2008), divide risk into two dimensions: probability (P) an impact (I). Risk (R) can mathematical be described as: R = P * I. Probability is derived from uncertainty of risk occurrence. The impact is the effect of the contingency, and therefore relates to the significance of the loss to the individual of organization.

In relation to probability, time is an important aspect. Time has only one direction (from present to future), and the more time there is between scheduling and the realization of the schedule the more chance there is that uncertainties become reality (as less information in known about the future state of the scheduling environment). The horizon used for scheduling, which is the amount of time a planner looks ahead, is therefore important. The smaller the planning horizon, the smaller the chance of uncertainties becoming real.

In this research the horizon is set to be one day: the detailed schedule for today is created and updated if necessary. This is referred to as *operational planning*. Other levels in planning are related to strategic decisions (i.e., long term) and tactical decisions (i.e., medium term). These decisions, for example related to the amount of resources and customer acceptance, are assumed as given. It is decided to focus on the operational planning as it is the type of planning that is dealt within GP, given the strategic and tactical settings.

2.2.1. Risk identification and classification

The framework from Schatterman, Herroelen, Van de Vonder and Boon (2008) is used to identify the risks for scheduling in the research context. Using their methodology, 15 sources of uncertainty are identified in this research. These risks relate to the environment, the organization, the product/service, workforce, machines, and materials. A list of these risks is given in Appendix C.

For each risk the probability and impact are estimated. Based on the estimations each risk is positioned in the probability-impact matrix. The probability-impact matrix is an often used (qualitative) method for assessing risks. In general it would also be good to analyse the risks based on quantitative measures, however in this research that is not useful as it would focus on a specific case.

Vasile et al. (2013) scored both evaluation criteria on a three-point scale (i.e., low, medium and high). Based on this scale, different sections in the probability-impact matrix are distinguished. Each section is assigned a rating (based on a five-point scale: insignificant, low, medium, high and very high). This rating indicates the importance for considering the risks. The probability-impact matrix is given in Table 2.1.

		Impact		
		Low	**Medium**	**High**
Probability	**Low**	-	Economic shifts Governmental restrictions Capabilities	-
	Medium	Weather influences	Inaccurate estimation of duration Urgency for completion	Availability of resources
	High	Traffic influences Indistinctness	Variation in prep. and cmpl. times	Variation in demand

Table 2.1 Probability-impact matrix of identified risks

Based on the matrix, most important risks are identified which are:

1) Variation in demand
2) Variation in preparation and completion times / reconfiguration times
3) Availability of resources

The second risk strongly relates to time. As mentioned before time is important in scheduling, but the main withdrawal of time is that it has only one direction. The aspect of time now becomes more emphatically present as a restriction in the scheduling function. In order to deal with those uncertainties, the scheduling function should adapt to the changing input characteristics. For that matter a dynamic scheduling engine is required.

The other two risk relate to the availability of input for the scheduling function. The variation in demand causes changes in what has to be scheduled, the changes in availability of resource limit the option of scheduling as the resource capacity fluctuate. How to deal with these three sources of uncertainty is discussed in Chapter 3.

2.3. Demarcation of input characteristics

While discussing about what characteristics are important in planning, some difficulties in the scheduling process are identified. A couple of difficulties are for example the dynamic arrival of events, the simultaneous requirement of multiple resources by one event, and scheduling sequential events (especially in relation to fixed resource routing). With the help of experts, a demarcation is made such that the general model provides a solid and representative basis. The basic model can later be expanded by considering more characteristics or constraints. This is done to stimulate the development of the basic automated planning process. Decisions and assumptions related to the basic model are made.

The following decisions are made:

I. The model focuses on one stage, single event scheduling. The relation to predecessor and successor events are indicated. However, the model considers each event as a standalone event, with individual release and due dates. The focus is not on optimal sequencing related events.
   By not particularly focusing on sequential events the main characteristic of events in the production environment is subordinated. However, the resulting model can be applied in all three selected business environments.
II. The model focuses on the assignment and sequencing of events on resources. It does assume that predefined routing is not applied.
III. The model focuses on short term scheduling: scheduling is done on daily basis. Long(er) term events, decisions, and influences are neglected.
IV. Events are assigned a predetermined duration. This implies that the event duration is independent of resource specific capacities.
V. Only regular resources are considered, this implies that outsourcing is not taken into account.
VI. General environmental factors, such as traffic intensity, are not taken into account as the primary focus is on the formulation of the basic planning problem. Factors like these might be valuable to consider in a later stage.
VII. Preparation and completion events are performed by the same resource that performs the main event. Preparation events are performed right before performing the main event. Completion events are performed right after performing the main event. Other required activities are assigned to another event.
VIII. Preparation and completion to-dos must be performed in the (percentage of) time that is reserved for them. Time to perform these activities is subtracted from the resource availability. The amount of time to reserve is set by initializing the resource utilization (in percentages).

The following assumptions are made:

I. Human resources are self-movable and can move independently.
II. The use of required not-consumable resources is indicated. However, picking/delivery/exchange of these resources by movable resources is not considered.
   An event requires human resources, and additional materials such as a stepladder. When scheduling the event, the required use of the stepladder is indicated. However, the availability and capacity of stepladders is not considered. In addition, picking up the stepladder at a central depot and bringing it back is also not considered.

2.4. Conclusion

The scheduling process always involves a set of resources and a set of events. Resources are some type of unit that are able to transform some type of input into a different output. Resources are characterised by their availability, capacity, utilization, skills and location. Events represent the jobs/orders that require action in order to be performed. Events are characterised by their type (and required skills), status, processing stage, release and due date, location and allowance of pre-emption.

In order to create a schedule, resources and events should be assigned to each other. This assignment is based on the required characteristics on the event side, and the deliverable characteristics on the resource side. However, matching those characteristics is complex as decisions are not univocal.

Even if a schedule is already created, there are sources of uncertainty that disturb the effectuation of the schedule. The main sources of uncertainty are variation in demand, variation in reconfiguration times, and changes in availability of resources. In order to deal with those changes a dynamic scheduling engine is required.

The high amount of characteristics and factors that influence the scheduling process makes it hard to create a scheduling algorithm that considers all factors and their (mutual dependent) relations. Therefore a demarcation of characteristics and factors is made. After the demarcation the following characteristics and factors are identified that must be considered in the scheduling algorithm:

- The availability of both resources and events
- The allowed utilization of resources
- The type and status of events
- The (required) capacity of both resources and events
- The (required) skills of both resources and events
- The system limitations on capacity based on utilization and availability
- The urgency to perform an event based on event type
- The time/costs/effort required for reconfiguration

Chapter 3 Literature on scheduling

This chapter focuses on the different solution models that are proposed in literature to solve scheduling problems that are similar to the research problem. In Section 3.1 commonly used scheduling problem formulations are described. The similarities and differences to the research problem are discussed. In Section 3.2 describes the methodology that is used to evaluated different solution approaches. Section 3.3 several solution approaches are described and analysed. The analysis provides insight in what kind of solution approach is useful for solving the research problem. Section 3.4 gives insight in risk management strategies that are used in scheduling. Section 3.5 captures preferred directives for a solution methodology. A summary of the results follows in Section 3.6.

3.1. Common scheduling problem formulations

When looking for dynamic scheduling problem solutions a large variety of solution is found. However, not all of them are useful and therefore a selection must be made. In order to structure the search for useful solutions several directives are used. A first directive is given by Billaut, Moukrim and Sanlaville (2008). They distinguish two major classes in scheduling solution methods: classical deterministic methods (i.e., offline methods), and online methods. The classical methods assume that data is deterministic, and the environment is relatively simple and stable. These methods are proactive, and applied before the realization of given activities. Online methods do not have all data at the start of the scheduling process and adapt to new information that occurs during the realization of the initial schedule. These methods are therefore reactive, and used when part of the schedule is already realized. The methodology of an online solution method can be very similar to a deterministic solution method. However, online solution methods often incorporate some slack to be able to deal with uncertainties and are required to be faster as they must be applied more often (i.e., in case of rescheduling). In the framework of this research an online approach is required, as the scheduling engine should be able to deal with the realization of uncertainties.

Although we now know how to approach the solution, the basis for a solution is a good problem formulation. A good problem formulation is the second directive to a useful solution. Therefore we will look at different scheduling problem formulations, and although they might not match exactly with the research problem as long as they are comparable (i.e., they do consider the same main characteristics as identified in Chapter 2) they provide good directions for a solution.

Based on a literature study, two formulations that have many similarities to the research problem are found: the vehicle routing problem (VRP) and the resource constrained scheduling problem (RCSP). In order to show those similarities, the found problem formulations are described in more detail in Section 3.1.1 and 3.1.2.

3.1.1. The vehicle routing problem

Laporte (1992) describes the VRP as the problem of designing optimal delivery or collection routes from one or several depots to a number of geographically scattered cities or customers, subjected to side constraints. The basic VRP consists of designing a set of least-cost vehicle routes such that each customer is visited exactly once by one vehicle, all vehicle routes start and end at the home location, and all side constraints are satisfied. Side constraints are for example restriction on capacity and time.

A variant of VRP is the traveling salesman problem (TSP) formulation. The basic TSP formulation deals with a salesman who is seeking for the shortest tour along *n* clients. All clients need to be visited exactly once, given a home location which is the fixed start- and endpoint of the salesman.

Within TSP the number of resources is given and so the focus is on finding routes for each resource while visiting all customers once. In a TSP the focus is on sequencing. Within VRP the number of resources is not given and so the focus is not only on sequencing, but also on clustering. Clustering is important in relation to the number of resources necessary. In addition to clustering decisions, also sequencing is important in order to find optimal routes. In this research the number of resources is (assumed to be) known and therefore a TSP formulation is more applicable compared to a VRP formulation.

The TSP has many variants, such as the multiple traveling salesman problem (mTSP). Within the mTSP a set of routes for *m* salesmen has to be found. All salesmen start from and return to a single home location (central depot). The mTSP has another expanded variant; the multi depot multiple traveling salesman problem (MmTSP), in which tours have to be found for *m* salesmen such that all customers are visited exactly once and all salesmen return to multiple fixed depots. There are two variants of the MmTSP; the fixed destination MmTSP where each salesman has to return to his original depot, and the non-fixed destination MmTSP where salesmen do not have to return to their original depots but the number of salesmen at each depot should remain the same (Kara & Bektas, 2006).

Table 3.1 Overview of TSP problem formulation characteristics

Problem formulation	Number of resources	Number of start locations	Fixed return depot
TSP	1	1	Yes
mTSP	>1	1	Yes
MmTSP fixed dest.	>1	>1	Yes
MmTSP non-fixed dest.	>1	>1	No

TSP formulations can be expanded by adding side constraints, such as a boundary on the number of customers on any route, time windows for visiting a customer, or precedence relations. By adding more elements to the model, the problem becomes more complex and often harder to solve (Surekha & Sumathi, 2011).

3.1.2. The resource constrained scheduling problem

Wall (1996) describes the RCSP as follows: “Given a set of activities, a set of resources, and a measurement of performance, what is the best way to assign the resources to the activities such that the performance is maximized?”. Ghallab, Nau and Traverso (2004) extended the model, by specifying a set of constraints on the activities and resources. This scheduling problem is often used when dealing with large scale projects. In relation to automated scheduling it is a problem formulation which is often referred to.

According to Geffner and Bonet (2013) there are three approaches that can be used to address the automated RCSP. First there is the *programming-based approach* in which the action to do next is given by the program. The program includes a collection of rules or behaviours. Second there is the *learning-based approach* in which the action to do next is induced from experience as in reinforcement learning. Third there is the *model-based approach* in which the action to do next is derived automatically from a

model of actions, sensors and goals. The programming-based approach is static and focuses on long-term patterns. The learning-based approach is partly static and partly dynamic, it does focus on medium-term patterns. The model-based approach is most dynamic as it makes decisions based on the current state of the system.

In addition to the problem formulation, Wall (1996) indicates that the objective(s) may be activity-based, resource-based, related to performance measures, or some combination of these. For the solution method there might be a difference between event-based or resource-based solutions. While using event-based solutions one assigns resources to activities (e.g., activities are considered as variables), while using resource-based solutions one assigns events to resources (e.g., resources are considered as variables). As both perspectives are important, both need to be partly integrated in order to find a feasible solution but the degree of integration depends on the selected solution method.

Wall (1996) recognizes that this is just a general problem description and many variations can be made. One thing that stands out in this formulation is that the solution method is resource-based: activities and resources are connected to each other based on resource characteristics. Although the characteristics of events are considered in their methods, their presence is subordinate to the characteristics of resources. In this research both resources and events are similar in importance, therefore we must carefully selected parts of the RCSP while focusing on a solution method for the research problem.

3.1.3. Preliminary conclusion on scheduling problem formulations

Although many variations of the TSP are described in literature, these models only partly describe the research problem. One of the reasons for that is that in this research a lot of factors need to be considered simultaneously. In literature the focus is often on single aspects, and so only clustering or sequencing is discussed in detail. Besides, the connection and integration between multiple factors is often neglected or not even considered. As we identified a large amount of influencing factors, the currently found models are not satisfying and therefore existing solution models should be expanded in order to fulfil all research problem requirements.

The RCSP is related to this research while the restrictions on capacity and availability are often constrained, but matching those specific characteristics on resources and events is an important aspect in the research problem but is less focused on in this problem formulation. In addition, the perspective in RCSP is from a resources' point of view. In the research the match between both resource and event is important, and within RCSP formulations the events' side is underexposed.

With the problem formulations as discussed in this section as inspiration, the research problem is formulated as follows:

The research problem deals with multiple resources who are all seeking for an optimal (i.e., most valuable) but mutually non-conflicting sequence of events. The assignment of events is limited by side constraints (e.g., skills and availability) and is further based on priority values that are related to other identified characteristics of resources, events and the combination of both. All events must be performed exactly once. All resource must start and finish considering their default settings (i.e., for human resources this is their home location).

Based on this problem formulation the search for solutions is started. For this matter solution methods used to solve the given theoretical problem formulations are used as a starting point. In order to enable ourselves to evaluate the performance of the different solution approaches later one, first a performance evaluation methodology is discussed. This is done in Section 3.2. Section 3.3 describes the different solution approaches.

3.2. Solution performance evaluation

The overall goal of any solution approach is to find a well-performing and feasible solution. However, there is no fixed definition of well-performing, but general algorithm performance measurements can be used in order to express the overall meaning of it. For that matter the overall evaluation criteria of Cordeau, Gendreau, Laporte, Potvin, and Semet (2002) are often used. They set four criteria to evaluate solution algorithms: accuracy, speed, simplicity and flexibility. Within these four criteria the abilities for software transferability and end-user adaption are also taken into account. These two aspects are often neglected in other evaluation methods. In order to evaluate an algorithm, the evaluation criteria are scored using a five-point scale (i.e., from very low to very high). The main benefit of this approach is that is can be used analysing all type of solution methodologies. This section describes the four evaluation criteria in more detail.

3.2.1. Accuracy:

Accuracy measures the degree to which the heuristic solution value deviates from the optimal value and is also related to consistency. Users generally prefer a heuristic that performs well given multiple occasions, rather than a heuristic whose performance fluctuates a lot (i.e., inconsistency). In addition to this, Cordeau et al. (2002) indicate that users prefer an algorithm that produces a good solution at an early state and increase the quality throughout the execution process, compared to an algorithm that requires more computational time and only returns a final answer.

The accuracy of a solution methodology is related to case specific performance measurements. These case specific performance measures represent the overall performance (i.e., efficiency) of a solution methodology. Efficiency relates to the use of the lowest amount of input in creating any given output (i.e., goals). How to measure the overall performance should clearly be defined at front. This in order to make sure that each reader has the same understanding. A good example for this is ‘workload’. The overall workload can be defined as the average number of events that is scheduled on a day, or as the percentage of time resources are not idle. In order to make sure that accuracy is a good directive for the quality of an algorithm, the clear definition (and understanding) of these case specific performance measures and their scale is important.

As the input and output differs per scheduling process, there is no single, specified objective that describes efficient scheduling. However, there is a common number of performance measurements that are used more often. Examples of such performance measurements are for example (Ramasesh, 1990):

I. Workload (average jobs per day)
II. Mean and variance resource idle time
III. Average number of jobs in queue
IV. Mean tardiness (+ number of jobs tardy)
V. Maximum lateness
VI. Average costs per job

A typical objective when striving for efficiency is to minimize the weighted combination of costs (i.e., based on a numerical scale or real costs) (Bard & Purnomo, 2005). This is especially useful when multiple characteristics must be considered. This method assigns a scalar value to each event (in combination with a resource and/or time) and schedules based on these values. The scalar values assigned to the different (costs) components are company specific, and so it can easily be customized. In order to use such a performance measurement in line with the method of Cordeau et al., the scalar value of the performance measure must be converted to a value on the evaluation scale (i.e., 5 point scale).

3.2.2. Speed

Speed relates to the computational speed of the algorithm and is also referred to as algorithm running time. The importance of speed depends on the required level of accuracy. When dealing with real-life problems the algorithm must be applied instantaneously in relation to changes. This criterion is very important in relation to solving the research problem.

The algorithm running time is often denoted with the Landau-notation, also referred to as big-O notation. This notation specifies a bound for the algorithm's performance. The expressed bound is an upper bound, as it gives the algorithm's worst performance. The actual performance for a specific input may be lower than the upper bound, but it never exceeds the upper bound. If the worst-case running time is expected to be larger than polynomial time (based on the number of calculation steps) it is hard to find the optimal solution within acceptable running times. The number of calculation steps that is can be performed in polynomial time equals $O(n^k)$ for some nonnegative integer k, where n is the complexity of the input.

The algorithm running time depends on the problem size, the software compiler, and the computer used. The software compiler and computer are assumed to be fixed, and so those cannot be affected by changing the solution algorithm. The problem size is defined by the number of variables and number of symbols needed to express the problem (i.e., combination and comparison of all variables and constraints that are required in order to form a decision).

3.2.3. Simplicity

Simplicity relates to the difficulty of understanding the reasoning of the algorithm. Algorithms that are difficult to understand due to any reason, such as containing too many parameters, are unlikely to be used. It is unrealistic to assume users understand all details of (scientific based) algorithms, but based on process knowledge the user should understand most basic solution steps.

In relation to scheduling problem, a minimum level of complexity is expected in order to capture the core of the problem and to return good results, but a simple code stand a better chance of being adapted. Therefore complexity and simplicity should be carefully balanced.

3.2.4. Flexibility

The flexibility of an algorithm relates to the degree in which the algorithm is able to handle with various side constraints, encountered in real-life setting, while still returning a feasible solution. General elements that influence the scheduling process are time, costs and required resources. As the scheduling engine is desired to be applied in multiple businesses, the scheduling decisions made by the scheduling

engine must be easily structured and so variables must be easily adaptable to customer settings. This implies that the boundaries of the automated scheduling algorithm must be flexible.

The flexibility of a solution is influenced by the constraints. Constraints are rules set for scheduling and limit the options. Constraints can relate to multiple aspects of the problem: it can be associated with a resource or an event, but it can also be related to resource allocation or time bounds. The identification of constraints is essential for the result of a solution approach while it affects the feasibility of a solution. This multitude of aspects for constraints makes the scheduling task challenging and difficult. Elements that are recognized as influences on the scheduling decisions are (Pinedo M. , 2005):

I. Time related issues; related to the release date, due date and duration of an activity, and available time-windows from both resource and activity
II. Capacity issues; related to the availability of resources
III. Resource issues; related to the configurations (e.g., capabilities) of resources
IV. Tooling and material issues; related to the required assets in order to perform a job
V. Workforce scheduling issues; related to the time schedule of individual employees
VI. Geographical issues; related to the operational area of a resource
VII. Costs related issues; for example related to lateness or denying customers

These four criteria are used to evaluate the performance of any solution approach. The best method scores well on all criteria, but often a compromise is necessary. In Chapter 1 it is said that the scheduling engine should provide good results. In terms of accuracy this implies that a certain threshold must be achieved. Chapter 1 also indicated that a solution must be created within reasonable time. This is due to the fact that we are dealing with real life problems and therefore a solution must be given (almost) instantaneously. Although it is not specified what is reasonable, we assume that a running time in second or maximum a couple of minutes is acceptable. For simplicity and flexibility no requirements are set in Chapter 1.

3.3. Solution approaches

The given theoretic problem formulation can be solved using *exact methods* or *heuristics.* Exact methods solve the given problem to optimality, heuristics are referred to as approximation algorithms and come up with a feasible solution that is not necessarily optimal. The size and the complexity of the problem determine which approach is best to be used. Small instances can often be solved to optimality relatively fast and so exact methods are applied; solving large instances to optimality is very hard (or sometimes practically impossible) and therefore heuristics are more suitable.

The decision on which solution methods to choose is mainly influenced by the required running time of an algorithm. Considering a basic TSP with n cities, the number of calculations in order to find the optimal solution is of the order of $(n\text{-}1)!$. In addition to this number of calculations, the running time of exact algorithms can increase due to added constraints. In this case the growth factor is usually higher than one. This means that if one constraint is added, the number of calculations increases with more than one (e.g., exponential or logarithmic growth) and so the proportional effect on the running time is larger. This quickly results in unacceptable running times. In relation to solvability and running times,

over a 100 real life problems of the traveling salesman problem are stored in a library referred to as TSPLIB. TSPs of all sizes are stored there, not all of them are solved. Within this library the largest TSP that is solved to optimality includes 85,900 locations in a VLSI application. However, solving this particular problem requires a central processing unit (CPU) time of 286 days (Cook, 2008). The running time of approximation algorithms are generally a lot smaller compared to the running time of exact algorithms, and are therefore useful while solving large problems.

Based on the size and complexity of the problem, it can often be estimated whether the problem can be solved to optimality or not. As mentioned before, considering the high amount of factors considered in this research, the problem is complex and also the size is relatively large. Therefore the focus is on heuristics. Choosing an heuristic a solution method still results in a wide range of (already designed) possible solutions, and so a more specific selection is required. Silver (2002) distinguishes two major classes in heuristics; basic heuristics and metaheuristics.

3.3.1. Basic heuristics

Basic heuristics are a powerful but often simplistic way to obtain a feasible solution (Silver, 2002). Silver distinguished multiple sub-classes of basic heuristics (not necessarily mutually exclusive) such as:

- *Randomly generated solutions*: randomly generates feasible solutions, evaluate each and select the best one.
- *Problem partitioning*: the original problem is decomposed into a number of presumably simpler to solve sub-problems. Each sub-problem is solved separately.
- *Constructive methods*: works by recursively constructing a set of objects from the smallest possible constituent parts. In each step, the best choice of that moment is selected by using a priority rule. This class is also referred to as greedy algorithms.
- *Improvement methods*: starts with a feasible solution to the problem (often the result of a constructive method) and evaluates neighbourhood solutions. Better neighbourhood solutions are accepted and iteratively evaluated.

In relation to scheduling constructive methods are most common. Examples of well-known constructive heuristic methods are for example, earliest due date scheduling, first in first out scheduling or nearest neighbour scheduling. The main advantage of constructive methods is their universal applicability, as they do not focus on a particularly niche. They are often used in ad-hoc networking and artificial intelligence. Within these fields, the creation of acceptable solutions within limited time is required. A constructive heuristic provides this. However, using such a heuristic may lead to worse long-term outcomes based on short-term solutions. When the result of the constructive heuristics is not satisfying (enough), improvement heuristics are often applied.

Improvement heuristics are mostly local and are executed by performing multiple iterations. Before an improvement heuristic can be applied, a (feasible) solution must be created (i.e., result of the constructive heuristics). Whether to apply an improvement heuristic depends on the quality of the result of the constructive heuristics. As indicated before (see Section 3.2.1) it is more favourable to produce an algorithm that return good quality solution at an early stage. Therefore the focus is on finding good constructive heuristics.

The common aspect in constructive heuristics is that they rely on myopic rules (Pinedo & Simchi-Levi, 1996). Myopic rules embrace two other types of rules: priority rules and dispatching rules. A priority rule is a function that assigns each waiting event a scalar value, and based on that scalar value events are selected over others for scheduling (i.e., prioritized). A dispatching rule dictates the distribution of orders over the available resources; it is a specific sequencing decision policy that defines the next activity of a resource. Priority rules therefore comprise a proper subset of dispatching rules (Gere, 1966).

Panwalkar and Iskander (1977) identified over a hundred different myopic scheduling rules. This is due to the fact that even a small change in the shape, size, or amount of available information in the problem formulation can result in a completely new solution formulation (Wall, 1996). To structure the different rules, they can be classified based on a number of attributes: first is it a singular or a combinatorial rule, second is it a static or a dynamic rule, and third is it a local or a global rule. Singular rules prioritize a single element, while combinatorial rules consider multiple elements and how they interact. Examples of some singular constructive dispatching rules are (based on Haugen & Hill, 1999; Panwalkar & Iskander, 1977; Gere, 1966) for example *earliest due date* (EDD) and *longest processing time* (LPT). EDD sequences orders based on their due date, the order that expires first is scheduled first. LPT sequences orders based on processing time, the order with the longest processing time is scheduled first. Singular rules are easy applicable and many of them are statically based.

Static rules are the ones in which the event priority value does not change as a function of time. Dynamic rules are the ones in which the job priority values do change as a function of the passage of time. An example of a dynamic rule is *nearest neighbour* (NN). NN schedules events based on their expected travel times. The travel time of resources depends on their current location. The priority value that expresses travel times therefore changes after each iteration.

Composite travel time expiration time (TTET) and *minimum stress insertion* (MSI) are examples of combinatorial and dynamic rules. TTET sequences orders based on travel times and system utilization. Orders with a short expiration time have priority if travel times are about the same, travel times become important when the systems have high utilization. MSI evaluates all possible insertions in all service employees' schedule and makes the insertion that adds the minimum stress at that time. System stress is the sum of the stress of all service employees. Stress of a service employee is the weighted sum of queue time, travel time, and tardiness for all service orders assigned to that service employee.

Local dispatching rules are those that require information only about the events that are in front of them at the same resource, while global dispatching rules require additional information about the events or the resource that are in the system. (Haupt, 1989)

In addition to those general rules, Gere (1966) describes multiple "tailor-made" approaches. Those "tailor-made" approaches are based on general priority rules, however, if the immediate situation calls for extraneous action, then exceptions to the general rule are allowed. *Alternate operion* (AO) and *time transcending schedules* (TTS) are examples of "tailor-made" approaches. TTS determines a priority rating for each event. The event with the highest priority value is selected for scheduling. The event is added at the most valuable moment into the schedule. This most valuable moment can be anywere in time, as long as there is a capacity available. AO determines a priority rating for each event, which can be based

on a single factor or on combinatorial factors. The event with top priority is schedules next. In addition to this rule, the priority of each event is re-evaluate after each iteration. If any other event reaches a critical level, revoke the last iteration. If scheduling the other event does not cause any event to be critical, whereas the first one did, then schedule the second event. Otherwise the first event is scheduled anyway. This procedure permits scheduling an event at any most valuable moment, and it allows one to evaluate critical decisions while applying the solution approach.

When applying a combinatorial rule the different characteristics are considered either sequential or simultaneous. Sequential rules consider each factor individual and based on each factor a sub-selection of events is made. Simultaneous rules consider all factors at the same time and assign a priority value for each evaluated event, each resource or each combination between an event and a resource. The only set requirement for a combinatorial dispatching rule is that it all different perspectives of the problem (i.e., it should cover elements of clustering (i.e., assignment criteria), routing, and sequencing) should be captured in that one rule.

However, the different characteristics are not necessarily expressed by the same scale (e.g., km, kg, and °C). To be able to compare the different factors and to see their (relative) importance, a common scale is required. For that matter a numerical scale (i.e., weights) or costs can be used. The advantage of weights over costs is that is does not require a measurement unit, only a value is enough. On the other hand, it is the main disadvantage as weights are abstract and so it can be hard to assign good values. The main disadvantage of costs is that some characteristics cannot easily be expressed in terms of money, an example of this is customer satisfaction or personal competences.

In relation to basis heuristics, there are also artificial intelligence (AI) approaches. Within AI approaches two classes are distinguished (Hildum, 1994). The first class contains expert systems, in which a heuristic approach produces a near-optimal solution by applying generic and domain-specific knowledge (e.g., situation-action rules). This class of approaches is not suitable for solving the research problem as for each case specific information is required, and so applying an expert system nullifies the general applicability of the model. The second class is pure knowledge based, and combines domain-independent knowledge with meta-level control knowledge. Most successful knowledge based approaches are: intelligent scheduling and information system, and opportunistic intelligent scheduler.

Intelligent scheduling and information system (ISIS) was the first generic, knowledge based system approach. The scheduling process is considered as a constraint-directed search of the space of possible schedules, and so it operates under a single, order-centred scheduling perspective. Orders are selected one by one based on their priority. The decision on what resource to schedule is made based on the attempt to satisfy all the constraints. With the help of a beam search, a sequence of events and resources is constructed for the event's developing process routing. Beam search is a technique for searching decision trees. The technique systematically develops a small number of solutions. By analysing most promising (partial) solution, one tries to maximize the probability of finding a good solution with minimal search effort (Peng Si Ow & Morton, 1988). The main disadvantages of this method is that its control is static and so the system lacks flexibility, and that the perspective is solely on events, and so it neglects resource-based conflicts. (Hildum, 1994)

O*pportunistic intelligent scheduler* (OPIS) consults both the order-based perspective, as well as the resource-based perspective while scheduling, and is an extension of ISIS. OPIS performs a capacity analysis based on the available data. If there is enough capacity (in a class of resource), a *resource scheduler* performs the task of scheduling (that particular class of resources). When new information enters the system, and the resource-based strategy is not required, an order-based scheduling strategy is applied (i.e., *ISIS scheduler*). The main disadvantage of OPIS is that is applicability to dynamic environments is questionable as the level of opportunism remains low, besides it falls short making effective use of the information given by the multi perspectives considered.

3.3.2. Metaheuristics

Silver (2002) describes metaheuristics as higher level procedures designed to guide other methods towards achieving reasonable solutions. The concept of metaheuristics is introduced by Glover (1986), and is inspired from analogies of natural processes. After their introduction the amount of metaheuristics developed rapidly. The application of some of the metaheuristics has been quite successful, although all their running times are quite large and their implementation can be quite complicated.

Metaheuristics can be divided into two categories: *single-solution methods* and *population metaheuristics*. Single-solution methods only consider a single solution (and search trajectory) at a time. Population methods evolve multiple solutions concurrently. Similar to basic heuristics, in each category construction as well as improvement metaheuristics can be distinguished. (Gendreau & Potvin, 2005)

Gendreau (2002) listed and evaluated some metaheuristics. He identified that local search metaheuristics (e.g., tabu-search, adaptive memory procedure and variable neighbourhood search) return good solutions quite rapidly, and that they are usually flexible enough to accommodate to specific complicating constraints that are present in real-life scheduling problems. Comparing metaheuristics gives results that are case specific. As developing, testing and implementing a metaheuristic for a very specific case is time consuming and costly, it is relatively hard to know what metaheuristic is best to use. Besides, the selection of a metaheuristic is highly subjective to personal preferences.

Although the selection of a metaheuristic is subjective to personal preferences, there are multiple heuristics that are more often referred to: tabu search, simulated annealing, adaptive neighbourhood search, genetic algorithms, and ant colony optimization. Vidal, Crianic, Gendreau, and Prins (2013) identified, classified, and analysed 64 (meta)heuristics on 15 different scheduling problem formulations, and concluded that tabu search, and genetic algorithms are more efficient compared to other meta heuristics.

Tabu search is a deterministic local search strategy where the best solution in the neighbourhood of the current solution is selected each iteration, even if this does not improve the current solution. A short-term memory, referred to a Tabu list, stores recently visited solutions (i.e., neighbours) to avoid short term cycling. This way, the method is able to escape from local optima. The search stops after a fixed number of iterations, or if the maximum number of consecutive iterations without improvement is reached. The implementation of tabu search can be quite cumbersome, due to the inclusion of many additional components which require a lot of parameters. (Gendreau & Potvin, 2005)

Genetic algorithms are part of the class of *evolutionary algorithms.* This methodology starts with a randomly or heuristically generated initial population. A renewal cycle is repeated for a number of iterations, and the best solution found is return at the end. The renewal cycle consists of three processes: selection, mating, and mutation. Within the selection process, the best solutions are allowed to become "parents". The mating process then takes two parents solutions and combines their most desirable features to create one or two offspring solutions. Each new offspring is subjected to small random perturbations e.g. mutation. The creation of offspring is repeated each iteration until a new generation is created. The best parent is selected at the end. Although this methodology is found effective, it often fails to generate near-optimal solutions.

3.3.3. Schedule generation schemes

The different classes of approximation algorithms are used in multiple situations, especially constructive heuristics, improvement heuristics and increasingly metaheuristics. In relation to scheduling, these types of heuristics are often applied while using schedule generation schemes (SGSs). SGSs are defined by Kolisch and Hartmann (1999). SGSs start with an empty schedule and build a feasible schedule by iteratively extending a partial schedule. Within a partial schedule only a subset of events is scheduled. In general there are two kinds of SGS: serial SGS which performs activity-incrementation, and parallel SGS which performs time-incrementation.

Basic heuristics are used on both of the SGS in order to construct a schedule. A priority rule is used for selecting the next event. When combining the priority rule based heuristic and SGS one can generate one schedule (i.e., *single pass method*) or more than one schedule (i.e., *multi pass method*). (Kolisch & Hartmann, 1999)

Generating a schedule based on some SGS and a priority rule, in combination with some improvement technique, often provides a good solution (Kolisch & Hartmann, 1999). However, there often is an optimality gap. Metaheuristics solve instances by exploring the solution space, and by doing so the often decrease the optimality gap compared to the use of approximation heuristics.

Indifferent of the selected class of heuristic, the decision to schedule serial or parallel is important. The resulting schedule can be classified into one of three types of schedules: semi-active schedules, active schedules or non-delay schedules. Semi-active schedules are feasible schedules obtained by sequencing events as early as possible. This implies that no event can be started earlier without changing the order of the events. Active schedules are the feasible schedules in which no events can start earlier without delaying some other event or violating preceding relationships. Non-delay schedules are feasible schedules in which no resource is kept idle when it could start scheduling an activity. Serial schemes always construct active schedules, while parallel schemes construct non-delay schedules.

Although the optimal solution is always part of the set of active schedules, but not necessarily part of the set of non-delay schedules, parallel schemes produce good average quality solutions. Several authors, such as Kolisch and Hartman (1999) and (Kim, 2009), have tested whether serial or parallel schemes provide better solutions in general. In both studies the serial SGS outperforms the parallel SGS, but Kolisch and Hartman indicates that there are a lot of cases (i.e., 59.73%) in which parallel SGS result in a similar performance.

Described applications of the SGS often relate only to one (type of) resource. When considering multiple resources, they are assumed to be homogeneous and represented as capacity. In cases where there are non-homogeneous resources, there are no examples found for the application of parallel SGS.

In addition, several authors such as Kim (2009), and Kolisch and Hartmann (1999) indicate that serial SGS outperforms the parallel SGS in several test cases (i.e., over 1000 projects distributed over 3 classes based on size). Kolisch and Hartmann (1999) additional indicate that priority rule based methods are indispensable when solving large problem instances in a short amount of time, and they provide initial solution(s) for metaheuristics procedures. They indicate that the development of good initial procedures is required before further effort is put into the development of higher level procedures. The focus is therefore on serial SGS, with a specific focus on basic (constructive) heuristics.

3.4. Risk management in scheduling

As indicated in the Chapter 2 there are source of uncertainty that influence the scheduling process. The difference in the output (i.e., schedule) of an offline and an online planning is due to the realization of some uncertainties. In reality a lot of companies perform short term operational scheduling activities in order to deal with changes and/or additions to the dataset, in other words they apply online scheduling. More important is that companies often create a scheduling environment that is somehow able to deal with those changes. This is an example of a risk management strategy.

Risks management strategies in general focus on reducing the consequences when risks occur. Specific for scheduling there are two aspects that is focused on while managing risks: *flexibility* and *robustness*. Flexibility refers to the degree of freedom during the implementation phase of the schedule (Billaut, Moukrim, & Sanlaville, 2008). Robustness refers to the ability of a schedule to remain stable under different scenarios (Mulvey, Vanderbei, & Zenios, 1995). In order to be able to make promises to customers and to be able to keep those promises, it is desired to have some degree of flexibility and robustness in the scheduling engine output. An indication for the required degree of flexibility and robustness can be made based on for example historical data analysis.

Flexibility and robustness are not a risk management strategy itself, but they are the result of it as they relate to the output of the scheduling process (i.e., the schedule). *Robust scheduling* is a proactive approach that yields solutions that are less sensitive to the model data and the realization of probabilistic information (Mulvey, Vanderbei, & Zenios, 1995). Mulvey et al. (1995) introduced the concept of robust optimization. They defined two distinct components that are present in optimization models: a *structural component* that is fixed and free of noise (i.e., disturbances), and a *control component* that is subjected to noisy input data. Next they define robust optimization problems by introducing a set of scenarios. Each scenario is associated with a set of realizations of control components. The solution can be robust with respect to the solution, and/or the model. S*olution robustness* is related to the optimality of realization, while *model robustness* is related to the feasibility of any realization. Within the context of the research problem both feasibility and optimality are important. In relation to customer satisfaction and process stability, also the feasibility of a solution is important. The number of changes that must be made due to the realization of uncertainties is desired to be low.

The modelling framework of Mulvey et al. (1995) does not indicate how to implement the different strategies into practice. A robust optimization method for scheduling is for example described by Xiaoxia, Janak, and Floudas (2004). Their methodology is applied to small scale mixed-integer problems. Uncertainties are translated into constraints, inequalities is those constraints must be equalized during the problem solving phase. The main disadvantage of this methodology is that it can hardly be applied on larger instances, and sources of uncertainty must be able to be translated into integer-formulated constraints. Based on the scale of the research problem, their methodology cannot be applied. Therefore robustness must be created using another methodology.

Manuj et al. (2008) listed eight risk management strategies: avoidance, postponement, speculation, hedging, control, sharing/transferring, and security. The different strategies are related to each other, and the use of one strategy entails the use of another.

- *Avoidance*: when working with a particular product or customer leads to unacceptable results, the specifications of those products or customers are avoided in general while selecting the business strategy on what to deliver and where to operate.
- *Postponement*: entails to delay the actual commitment of resources to a specific task to maintain flexibility
- *Speculation*: making decisions based on anticipated (customer) demand
- *Hedging*: the likelihood to be affected by the risks is estimated based on statistic data. The portfolio of resources and events is composed in such a way that not all entities are affected at the same time/with the same magnitude.
- *Control*: incite vertical integration by reducing risks for supply and demand in the supply chain.
- *Transferring*: achieved through outsourcing, off-shoring, and contracting.
- *Security*: identify unusual or suspicious elements and focus on them, to enable one to deal with the rest of the movements through a sampled-based process.

The different risk management strategies are separately or combined used in different practiced solution approaches. For example it is used by Hans, Wullink, Van Houdenhoven, and Kazemier (2008). They studied scheduling surgeries in operating rooms, where the occurrence of emergency surgeries is uncertain. However, their occurrence can be estimated and the operating room capacity is corrected for these omitted surgeries. In order to deal with the uncertainty related to the duration of surgeries, slack is planned on each day. Slack is based on the expected variance of the durations of the surgeries planned, and should prevent for working in overtime. Their methodology plans elective surgeries based on a constructive heuristic, which they improve by using a local search method. An (undescribed) insertion heuristic is used to schedule emergency surgeries in the online schedule. In this case speculation is practiced as main risk management strategy.

The study of Hans et al. (2008) relates to a study of Wullink, Van Houdenhoven, Hans, Van Oostrum, Van Der Lans, and Kazemier (2007), where they analysed two approaches on how to reserve capacity for omitted surgeries. The first approach concentrated all reserved capacity on dedicated resources; the second approach considered evenly divided reserved capacity over all resources. The second approach led to major improvements on the evaluation criteria (i.e., waiting time, working overtime, and overall

resource utilisation). A major benefit of this approach is that it can be applied on both small and large scale problems. This risk management methodology deals with risk similar to the risks identified in this research, and can therefore easily be adapted.

For the identified risks we defined the following strategies, based on best practice (Manuj & Mentzer, 2008):

1) *Variation in demand* *(**very high** risk)*
   This risk is very important and can easily disrupt the scheduling process. Postponement and speculation are often used in practice. The more one knows about what to schedule, the better one can make accurate scheduling decisions. The other risks management strategies are less useful as they cannot be applied without drastically change the environment.
2) *Availability* *(**high** risk)*
   Unavailability of resources is undesired and can best be avoided. However, for example illness of employees cannot be avoided. If not possible to avoid the risk, it restricts the scheduling options. The main alternative is to control the (un)availability of resources.
3) *Variation in preparation and completion times* *(**high** risk)*
   Speculation and hedging are often used to deal with those durations. Estimations are made for the duration which is based on for example historical data. The estimated durations are used for offline scheduling. For online scheduling, security is added as risk management strategy. By paying attention to disruptions, deviations can be observed relatively fast and so actions can be taken that prevents the schedule from larger consequences.

3.5. Directives for a solution

Although three core decisions are already made (i.e., the focus is on basic heuristics, the application focuses on serial SGS building, and a model-based approach is used) there is still a large amount of heuristic solution algorithms available. However, no specific instance is found that matches on all aspects with the research problem. Therefore the focus shifts to a combined solution approach or even a customized solution algorithm. In order to come up with such a solution, decisions are made about the design of the identified attributes: multiplicity of input characteristics (i.e., singular or combinatorial rule), dynamics (i.e., static or dynamic rule), and the scope (i.e., local or global rule). Based on the advantages and disadvantages of the functioning of those attributes as found in literature, we can make decisions which we will use to structure a solution methodology.

3.5.1. Multiplicity of characteristics

The multiplicity of input characteristics is clearly more than one, as there as multiple factors identified that characterizes an event. However, one can decide to combine those factors and then work with an overall priority value, or one can decide to sequentially consider each individual factor.

Whether an overall priority value per resource-event combination is better than the sequential considering different characteristics of both resources and events depends on the characteristics. The first option is very general and does not exclude any option prior to its evaluation. The second option creates a sub-set, and so the succeeding number of calculations required is smaller. For example if one is

scheduling for a certain day, based on the availability of any event of that particular day a specific sub-set of events can be made. Events that are not available are not included in the sub-set and therefore they cannot be scheduled. Considering another characteristic, such as travel distance, it might not be smart to create a sub-set. Let us say that one starts at a central point and can travel in any direction. At first, based on distance it might be best to select an event that is close. Travelling in the required direction for that event, for example east, will enlarge the travel distance to other directions, such as west. Besides it will decrease the residual distance further to the east. Creating a sub-set based on a maximum travel distance might exclude events that are good options as resource characteristics, such as location, changes.

However not excluding any event, on for example travel distance, will not decrease the number of options and so the solution space remains the same. If the number of possible events is large, from a computational point of view it is wise to exclude some events. Otherwise all options must be reconsidered each time which will require many calculations.

Another element that must be taken into account while selecting a solution approach is that the boundaries of the algorithm are flexible, such that customization is possible. This directs to a combinatorial scheduling rule. Within such rule the previously identified factors are considered, however their relative importance is not predetermined. When applying a combinatorial rule, the importance is variable. This enables one to apply a basic combinatorial rule that is able to satisfy specific customer needs. The only set requirement for the combinatorial dispatching rule is that it must include all different perspectives of the problem (i.e., it should cover elements of clustering (i.e., assignment criteria), routing, and sequencing).

Based on the size of the set of events and the variability of the related input or resource characteristics related to a specific event characteristic, it is wise to decide how much to limit the solution space using sequential evaluation of characteristics.

Limitation of the original set of events to a reasonable set of events, depends on the number of available resources and the number of events expected in the final schedule. A reasonable set of events contains at least the maximum expected number of events in the final schedule plus some surplus (i.e., 10 percent). When estimating the maximum expected number of events, the effects of the realization of uncertainties must be considered.

3.5.2. Static vs. dynamic decisions

In relation to the research problem, and scheduling in general, locations (in the service and sales environment) and times are expected to change drastically. Changes in time-related elements have a great influence on the availability of events, and so it has a major influence on the feasibility of a schedule. Changes in locations (i.e., travel times) have a great influence on the required preparation times. As preparation times influence the selection decision for scheduling. Both types of changes are expected to have a major influence on the schedule, and therefore some adaptations must be made during the scheduling process.

The dynamics that must be considered are partly discussed in the previous sub-section, for example the location of a resource, or the time an event can be scheduled. These characteristics change during the scheduling process and it is required to adapt to these changes while practicing the solution algorithm, otherwise it results in a poor or even non-feasible schedule. Therefore a dynamic rule is expected to be more valuable than a static one.

However, if it takes more effort to update the characteristics and adapt to the new situation compared to the original shift on the characteristic, adaptation is not recommended. Whether the solution algorithm should adapt to (expected) changes in the decision characteristics should be determined based on the impact on the final schedule. It can be measured in terms of accuracy and speed. Using a static rule generally requires fewer calculations, and therefore the algorithm is faster. However, the expected changes can negatively influence the accuracy, and even the feasibility, of the schedule.

Overall the changes in for example locations and preparation times must be considered. Neglecting those changes will result in very poor schedules. The impact on the accuracy is therefore overruling the increase in speed. Therefore a dynamic constructive heuristic is recommended, as long as the impact on speed weights up to the realized increase in accuracy. However, the impact of neglecting the changing characteristics must be tested.

3.5.3. Local vs. global decisions

A directive within heuristics is derived from the approaches as proposed by Geffner and Bonet (2013). They proposed three approaches that can be used for automated planning (see Section 3.1.2). From these approaches, the model-based approach fits best with what is desired, as it induces actions based on sensors and goals. This methodology goes beyond the scope of the other approaches, and the current scope of a lot of planners. When using this approach, automated scheduling is a step forward, and not just a capturing of the knowledge of planners.

The final result, i.e., the generated schedule, is most important. As indicated before, the result should be analysed based on some performance measures. These measures assign some value to the generated schedule.

When working with large problems, heuristics are often divided into multiple stages. Generally there is a clustering/assignment phase, and a sequencing phase. As an additional phase, there might be an optimization phase. An often used heuristic that uses two phases is the *cluster first, route second* algorithm (for example Gere, 1966 and Giosa, Tansini, & Viera, 2002 described several multi-phase algorithms). This is an example of an algorithm that uses a local rule, as the problem is solved for each cluster.

The main disadvantage of local rules is that they do not consider the overall goal and/or the long term effects. Local rules often result in a local optimum. Even if there is an additional third phase that focuses on improvement, it is hard to escape from a local optimum. (Gere, 1966)

Another reason to prefer a global rule over a local rule is that the research problem deals with a continue scheduling process. The long term position of the organization is influenced by the decisions made in scheduling. Therefore it is important to keep a close eye on the long term performance.

3.6. Conclusion of proposed solution methodologies

Two common used problem formulations for scheduling are discussed. Each of them has similarities and differences to the research problem. The differences are mainly caused by the large amount of factors considered in this research, and the requirement that the automated scheduling function is applicable in multiple businesses within multiple environments. This implies that the solution method must be able to be customized, considering the input characteristics, but also considering the objective.

A second point that came forward is that in automated planning one generally seeks for good and feasible plans rather than optimal plans while similar combinatorial optimization problems cannot be solved to optimality within polynomial time. Therefore the focus is on heuristics, instead of exact methods.

Two major classes of heuristics are distinguished; basic heuristics and metaheuristics. The main difference is their running time. Basic heuristics use myopic rules to construct a solution and provide acceptable solutions within limited time, and they can (relatively easily) be adapted to a universal, as well as a specific, problem formulation. Metaheuristics are harder to understand as they are often derived from nature processes, and they require more running time. Although metaheuristics have proven that they can be quite successful, their running time and often very complex implementation trajectory made us decide to exclude them. Therefore the search for a suitable solution method is on basic heuristics.

In addition to the type of heuristic used, the type of scheduling must be considered. There are two types of schedule generation schemes: serial and parallel. When using the serial SGS, the number of stages equals the number of activities to be scheduled. In each stage, one activity is selected to be scheduled. The activity is selected based on a priority rule, and then scheduled at the earliest precedent and resource feasible completion time. The parallel SGS divides a timespan in multiple stages (i.e., timeslots). Based on the scheduling time, a set of activities is selected. From this set of activities, most valuable activity is selected to be scheduled. The value of an activity is based on a priority value, which is based on a mathematical representation of characteristics. Kolisch and Hartmann (1999) indicate that serial SGS outperforms the parallel SGS in several test cases, and also that serial SGS are easier to use to improvements. Therefore the focus is on serial SGS.

While focusing on heuristics, there is no heuristic found that matches on all aspects with the research problem. Therefore a customized algorithm seems logical. The following guidelines are set for the design of such an algorithm: it should be a dynamic and global rule that balances sequential selection and combined optimization is desired.

Based on the evaluation criteria of accuracy, speed, simplicity and flexibility, a constructive heuristic solution is most logical as this type of solution approach scores well on all aspects. The accuracy and speed will be determined by the design of the algorithm. The simplicity of constructive heuristics

supports acceptance of the solution. Flexibility is important as the solution must be general applicable and accommodating different customers. In addition to the constructive heuristics an improvement heuristic can be used. In this case a two-phase heuristic is created. Two phase heuristics can deal with uncertainties relatively easy. However, first the focus is on creating a solid constructive heuristic.

An adapted greedy algorithm can therefore best be designed. A greedy algorithm works by recursively constructing a set of objects from the smallest possible constituent parts. Regular greedy algorithms do not necessary focus on the overall performance, but this focus should be merged into the solution algorithm.

In addition the greedy algorithm should be extended with a risk management strategy. For that matter the solution methodology of Hans et al. (2008) can best be used. The solution methodology must be adapted at some points in order to merge into the algorithm but it already does consider most identified risks. The new solution algorithm then comprises two algorithms. The first algorithm creates an initial schedule based on the information that is already known but it should leave space for the expected realization of risks. A second algorithm is used to adapt to new information. The second algorithm is for example used when urgent events occur that must be fit into the current schedule. The second algorithm should evaluate the options for adding the event to the existing schedule. The design of the second algorithm can be very similar to the first one, however its applicable window should be larger. For example, the first algorithm limits scheduling up to 70 percent of the total capacity. While using the second algorithm this limitation should be removed. However, first a good and stable first algorithm should be created.

Chapter 4 Solution methodology input variables

As the main directives for a solution methodology are known, a basic design can be created for the research solution methodology. The key directives point to a constructive heuristic, i.e., greedy heuristic. The main benefit of using a greedy algorithm is that is can easily be customized and changed.

A general greedy algorithm adds the best choice of that moment to the schedule. To decide what event, or what combination between event and resource is the best choice, is determined by using a priority rule. The priority value of an event is based on up to date characteristics and therefore dynamic. Characteristics are related to both resources and event and their contents might change over time. Effort is required in order to match the characteristics on both sides. However there is a static part in the overall priority value of a combination, which is related to those characteristics that do not change over time. The characteristics considered are derived from Chapter 2, and if necessary divided into more detailed elements

In section 4.1 it is decided on what characteristics influence the priority value of an event or a combination of event and resource, and it is explained how they are primarily considered. Section 4.2 is dedicated on restriction design. Both the priority value and the restrictions are used as input for the greedy scheduling algorithm as proposed in Chapter 5.

4.1. Priority indication

The characteristics that determine the static value of an event are the so-called "*which* factors", as they are related to the question of which events must be scheduled. Characteristics that influence the priority value of an event, as identified and delimited in cooperation with experts, are:

I. Event type
   Performing an urgent event adds more value compared to performing a regular event. Performing an event with higher required service performance agreements should therefore be preferred over other event types.
II. Waiting time of the event
   An event that is already waiting a couple of days becomes more important as the company wants to satisfy all their customers. This prevents relatively unimportant events from being unscheduled.
III. Time left till due date
   An event must be performed before its due date, otherwise penalty costs occur. For this matter the time left until the due date is an important aspect. Events close to their due date are more likely to be scheduled.
IV. Agreement on timeframe
   When promised to a customer to perform a certain event on a certain day within a certain timeframe it is undesired to change that decision. Rescheduling this particular event must be prevented.

Knowing what event has the highest initial total value does not indicate to what resource the event is assigned. Therefore "*where* factors" must be included, the influence of these characteristics on the

priority value is affected by the decision what resource to match with what event. The following characteristics, related to both resource and event, influence the total value of a combination:

V. Being a preferred resource
Customers might favour a certain resource based on experience or company specific operational site certificates. Assigning these resources to the event adds more value compared to the assignment of another resource.

VI. Variable preparation costs
Preparation activities involve time and costs, and depend on the current state of the assigned resource. For example a stove that needs to be heated requires more time to get from 100°C to 200°C than from 100°C to 150°C, therefore based on preparation time it is wise to schedule the event that requires the 150°C first. However, if the current state was not 100°C but 175°C, the difference in preparation would be indifferent.

VII. Variable completion costs
Completion activities also involve time and costs, and depend on the current state of the assigned resource. The same reasoning can be applied as for 'variable preparation costs'.

Next, there are two characteristics, related to the assigned timeframe of an event on a resource, that also influence the priority value of an event. These characteristics are referred to as "*when* factors" as they result from a specific combination of resources and events in addition combined with a specific time.

VIII. Costs for working outside of customer timeframes
It might occur that one expects an event to finish outside the availability timeframe of the customer. It must be decided whether this is allowed (with given boundaries) or not. If allowed, costs are involved for working outside these timeframes as it requires the customer to adapt.

IX. Costs for working outside of regular working hours
The same reasoning can be applied for the working hours of employees. It might happen that an event must be dealt with outside regular working hours. For the time worked outside of regular working hours additional costs are involved.

4.1.1. Calculation of the priority value

Each characteristic identified above should match between event and resource. In order to do so changes are required (e.g., heating a machine). The effort required to match those characteristics is expressed by using a scalar value that is assigned combination between an event and resource. A combination of each "*which-where-when"* combination results in a value that represents the overall value of the particular event on a particular resource at a particular time, and therefore it is a static value. The goal is to create a final schedule where the overall sum of priority values related to all scheduled combinations is the maximum.

The scalar value assigned to each characteristic should contribute in the direction they influence the decision. Changes in the content of characteristics that positively influence the decision should have a positive value, factors that negatively influence the decision should have a negative value. Most valuable

combination should be selected for scheduling, while the goal is to maximize the total added value. Some characteristics must be considered per unit, for example cost per hour or costs per kilometre.

The priority value of a combination between an event and a resource is calculated by addressing the characteristics of both event and resource and their combination. Figure 4.1 gives an example of two events that can be scheduled. Figure 4.2 gives an example of an available resource. Based on the priority values (see Figure 4.3) the priority values of both combinations can be calculated.

Event 1

Type: urgent
Status: released
Required materials: none
Release date: 19-01-2016
Due date: 22-01-2016
Number of required resources: 1
Plangroup: 1
Resource group: 1
Expected duration: 1hour
Required skills: 1
Preferred resource: B
Available: 8 – 17 each working day
Location: 10,10 (based on x,y indication)
No preparation/completion events

Event 2

Type: regular
Status: released
Required materials: none
Release date: 01-01-2016
Due date: 31-01-2016
Number of required resources: 1
Plangroup: 1
Resource group: 1
Expected duration: 1hour
Required skills: 1
Preferred resource: B
Available: 8 – 17 each working day
Location: 5,8 (based on x,y indication)
No preparation/completion events

Figure 4.1 Example events chapter 4

Resource A

Plangroup: 1
Resource group: 1
Availability: 8 – 17 each working day
Capacity/utilization: 90% of 8h per day
Skills: 1 and 2
Home location: 0,0

Figure 4.2 Example resource chapter 4

Assume that the time the event is being scheduled has no influence here, the current priority value of this combination can be calculated based on predetermined weights. The next box gives some example weights. After that the priority value for this resource-event combination is shown.

Weights: Date : 20-01-2016
- Event type:
  a) regular = 100
  b) urgent = 200
- Waiting days = 5 per day
- Days till due date = - 5 per day
- Being a preferred resource = 40
- Preparation/completion per unit = -7,5 per travel distance unit
- Penalty costs working outside timeframe = -25 per hour
- Cost for outsourcing = -10 – (5 per hour)

Figure 4.3 Example priority weights

Based on their characteristics such as resource group, plan group, etcetera, the resource can be matched to both events. The priority value for each combination is shown in Table 4.1.

	Event 1	Event 2
Static		
Event type	200p	100p
Waiting days	1day = 5p	19days = 95p
Due date	2days = -10p	10days = -50p
Dynamic	Static value = 195p	Static value = 145p
Preferred resource	0p	0p
*Preparation (based on distance)**	20km = -150p	13km = -97,5p
Penalty costs	0p	0p
*Costs for outsourcing***	-15p	-15p
Total value	30p	32,5p

Table 4.1 Event priority value example

*Distance in the example is based on Manhattan distances. Also straight distances can be used in theory. In reality it is best to estimate the travel distance/duration as accurate as possible. More about this in Section 6.1.
** It is assumed that outsourcing is not required in the example.

Based on the static value event 1 is the best event to be scheduled on resource A. Based on the dynamic value it is best to schedule event 2 first on resource A. However, using the same reasoning the priority value for each combination at each moment in time can be determined. This example shows that it makes a difference whether one schedules based on static or dynamic value.

4.2. Restriction design

Restrictions limit the options for scheduling. This applies when, due to some limitation, a certain *which-when-where* combination cannot be scheduled. The way restrictions are considered in the scheduling is not predetermined and can differ per company, the design of constraints is due to these reasons often neglected in performed research. This is done although general guidelines can relatively easily be stated. In most research cases a lot of assumptions are made, or the representation of constraints is simplified. However, constraints have a large impact on the solution space (Wall, 1996), and so the decisions about their design are important.

Based on the characteristics of both resources and events, a number of constraints can be drawn relatively easy. For example, to be able to assign an event to a resource, both should have/require the same resource group, and the same skills. These constraints specify *which* things can be done, or *where* actions must be performed.

There are constraints that are more complex. These constraints are related to time and so they relate to *when* things are done. The allowance to schedule some event can therefore fluctuate and must be reconsidered during the scheduling process.

4.2.1. "Which and where" restrictions

This section describes the "which and where" restrictions. These restrictions determine what events can be scheduled, where they can be scheduled. They limited the possibilities of events to be scheduled on any resource. Unless mentioned otherwise, these restrictions are hard restrictions and so they must be satisfied.

4.2.1.1. Event status

The status of an event allows an event to be scheduled or not. Only events with an event status that indicates that the event is released to the system can be scheduled. For example, an event that is currently being processed with status “in progress” cannot be rescheduled without consequences.

The availability of materials can be related to the status of an event. For example, if the required material assets in order to perform a job are not available, the event status can be marked as “insufficient”. This identification signals the system user that the current assets are not fulfilling the needs and that some assets need to be purchased.

Pre-emption of events

Pre-emption of events is not allowed. Once an event is started, the event must be performed completely. If an event cannot be finished due to missing materials or anything similar, the current event is closed and a new event is created (i.e., a continuation event). The new event is assigned an event status on which it can be scheduled. It is favourable to schedule a continuation event relatively fast, as it is undesired to keep a customer that should already be dealt with waiting. This is captured in the release and due date assigned to the continuation event.

4.2.1.2. Matching requirements of event and resources

The assigned resources to an event should meet the required characteristics. This is applicable for the following characteristics: resource group, plan group, skills, and preferred resources. The availability of both event and resource should also match, but the availability is influenced by time and therefore discussed later.

Matching resource/plan group

Matching events to a specific resource group or plan group is a hard restriction, and therefore an easy yes-or-no question. If an event requires a resource from a specific resource or plan group, this requirement must be satisfied. For each combination this match must be checked before calculating the priority value of the combination, if there is no match the priority value does not have to be calculated.

Matching to a preferred resource

Matching resource requirements has also been an easy yes-or-no question so far. However, the question raises what to do if there are multiple matches between an event and the resources. It might be the case that an event has a *preferred resource*. One can decide to formulate the selection of a preferred resource as a hard restriction (i.e., the event must be assigned to the preferred resource), but it is more likely to be scheduled as a soft restriction. Hard restrictions are those restrictions that must be satisfied; soft restrictions are those that are desired to be satisfied. A preferred resource is preferred over others that have the same matching characteristics. However, the preference for a particular resource is not based on all resource characteristics. Therefore the selection of a preferred resource can better be expressed as a soft restriction.

This is illustrated by Figure 4.4: resource 1 is preferred, but resource 2 is closer. The needed preparation time now influences the decision on what resource to schedule the event. The preference for resource 1 must compensate the additional distance, otherwise resource 2 is selected. The preference for a specific

resource must therefore be expressed in relation to other resource characteristics. This can also be done by assigning a weight to it.

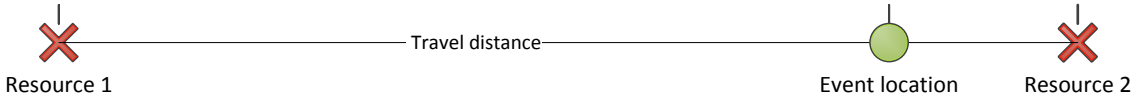


Figure 4.4 Resource preference

Matching based on skills

Matching based on skills is original also a yes-or-no question, but can conditionally be expanded. For example, event A requires skills 1 and based on all other characteristics both resource 1 and 2 are eligible to perform event A. Resource 1 only possess skill 1 while resource 2 possess skill 1 and 2. Scheduling event A on resource 1 result in more options later on as resource 2 is still available and able to process more diverse events. Therefore a conditional weight on skills is convenient.

4.2.1.3. Availability

The availability of both resources and events is bounded by timeframes, which are based on for example working hours and customer opening hours. In addition availability can be limited by time restrictions based on for example laws (e.g., time restrictions due to required breaks).

Resource specific timeframes

The scheduling engine should consider only the timeframes resources are available. The availability of resources is limited by utilization restrictions, which are for example stated by laws. Figure 4.5 displays a roster that keeps regular available hours of multiple resources in a (relatively) easy way.

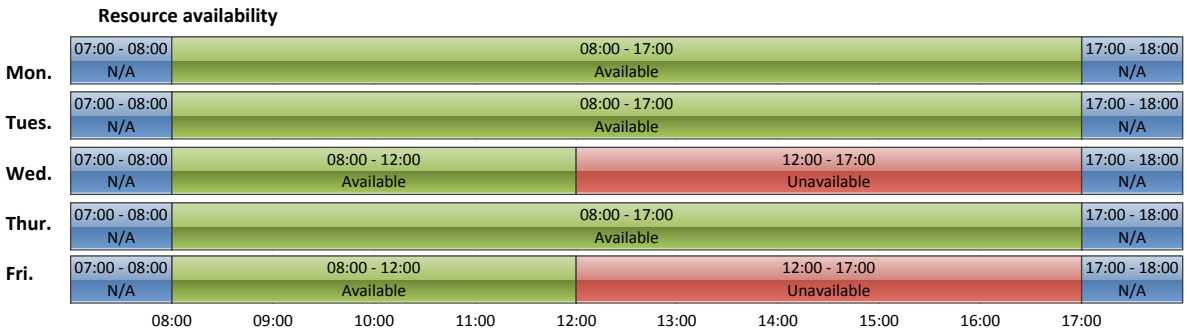


Figure 4.5 Regular resource specific working hours

In addition to regular resource specific hours, resources might be unavailable on irregular basis. Irregular unavailability can be expressed in two forms: planned irregular unavailability (e.g., day off, a doctor visit, scheduled maintenance) and unplanned irregular unavailability (e.g., illness, machine failure). In contrast to regular resource specific hours, dealing with irregular unavailability is complex as their appearance and duration is uncertain.

In the current system, planned irregular unavailability is dealt with by assigning “indirect tasks”. Using “indirect tasks” is an organized way to structure irregular unavailability. These indirect tasks are fixed on resource and time, and have strict start- and end-times. This methodology works for scheduling planned irregular unavailability as this type of unavailability is generally known before creating the schedule.

However it does not work for unplanned irregular unavailability, as type this of unavailability occurs when the schedule is already created.

In order to deal with unplanned irregular unavailability, one should consider the probability and the impact of unplanned irregular unavailability. The probability of the occurrence of major unplanned irregular availability changes is low: the chance of a resource being ill is small. However, if known that someone is ill today the probability that he is ill tomorrow is significantly higher. The impact of major availability changes depends on the size of pool of resources: while having 10 resources in a plangroup, the absence of a resource is relatively small. However, if there were just 2 resources the impact is high. As the probability is low and the impact is mediocre (based on scale of research problem), one can better accept the risk of the realization of unplanned irregular unavailability. If the impact or probability increases one should act proactive, which can for example be done by adjusting the availabilities in advance. When dealing with major changes in the availability of resource, the scheduling algorithm can best be applied again with adapted resource availability.

Customer specific timeframes: release and due dates

The availability of an event can be represented in the same way the availability of resources is presented. However, resources are generally available over a longer period and therefore their regular availability can be kept in a roster. The availability of an event depends on their release and due date. These dates are related to service level agreements (SLAs). Within an SLA the scope, quality, frequency, and responsibilities of an executor, in relation to its service, are defined. By creating a SLA, a customer knows what to expect from the executor, and the executor knows what promises they have to keep.

An SLA statement is for example that a specified customer is visited twice a year. For both visits an event is created. It is desired to plan these events about six months apart from each other. However, there are two options for scheduling these events. The first option determines the date for the second event based on the original scheduled date of the first event. The second option determines the date for the second visit based on the execution date of the first event. This decision is often made based on the contract, and should both be able to be merged into the scheduling algorithm. Let us assume an event can be scheduled within a time interval of a month around its ideal scheduling day (see Figure 4.6).

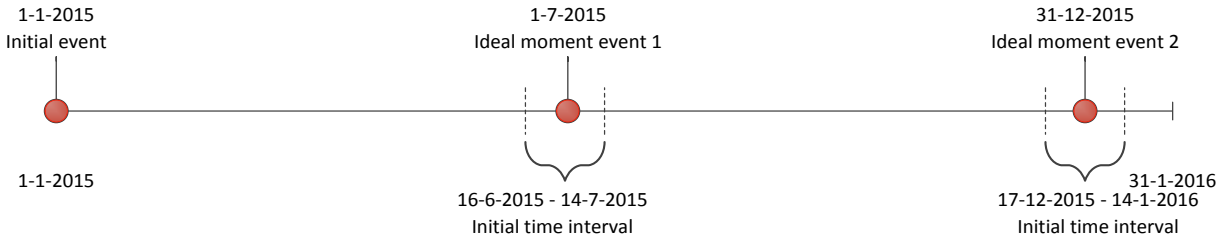


Figure 4.6 SLA based event scheduling

All events must be scheduling within their given timeframe that start at the release date and end at the due date. The time between the release date and the current date, as well as the time between the current date and the due date must be considered while scheduling. Customers that are waiting longer are more favourable to visit, as well as customers that must be visited within a limited time.

Regulation scheduling

Within the timeframe of regular working hours, worktime regulations might be applied. Some of these regulations might require a specific time within a timeframe. For example, when working from 8AM to 5PM a break of 30 minutes is required and must be scheduled between 12PM and 1PM. Although there are several ways to introduce such regulations into the schedule, most understandable way is to extend the duration of the event that is scheduled at 12PM. For all regulations such controls must be built in into the scheduling algorithm.

Outsourcing

Some companies have the ability to outsource the performance of events in case their regular capacity is too low to handle all necessary events. The resources related to outsourcing are undesired to be used if not necessary. The scheduling engine can be expanded with those additional resources. The use of those resources should be assigned a negative weight, and regular resources are always preferred. These additional resources are initially left out of scope.

4.2.1.4. Capacity and utilization

Capacity can be viewed at from two perspectives: the resource perspective and the system perspective. The capacity of a resource is the number of minor time units a resource is available during major time units. For example, on a regular working day (i.e., major time unit) a resource is available from 8AM to 5PM. The capacity is 9 hours (i.e., minor time unit). The sum of the available hours of all resources per time unit is the total resource capacity (i.e., system perspective). The total resource capacity can be used as indicator for the amount of events that can be planned in the given timeframe.

In order to create robustness, it is previous decided to schedule according to the methodology used by Hans et al. (2008) where capacity is reserved in order to deal with emerging events. Based on historical data analysis, the amount of time dedicated to deal with emerging events is determined. A reason to use this methodology is that it is easy to understand, and applicable on both small and large size systems.

By planning according to this methodology, one should think of how to divide reserved capacity over the planning horizon. The division of reserved capacity must be done in line with the expected realization of uncertainties. By doing this one tries to minimize the gap between the offline schedule and the online schedule. This can be done by splitting the planning horizon into time windows, for example split a day into morning and afternoon. Although it now might seem logical to divide the reserved capacity as a ratio over the available time windows (i.e., assign half of the reserved capacity to each of the time windows), one should realize that this brings another source of uncertainty as emerging events might not occur evenly spread over the planning horizon. Besides that, the input characteristics for the second timeframe are unknown when creating the schedule for the whole day.

One may for example conclude that 30% of the total resource capacity is dedicated to perform ad-hoc events. Let us assumes that there are five resources (each available from 8AM to 5PM), then the total resource capacity is 45 hours whereof 13.5 hours must be reserved. Even if there are multiple timeframes, and/or resources are not fully available, the reserved capacity is equally divided and result in a schedule as in Figure 4.7.

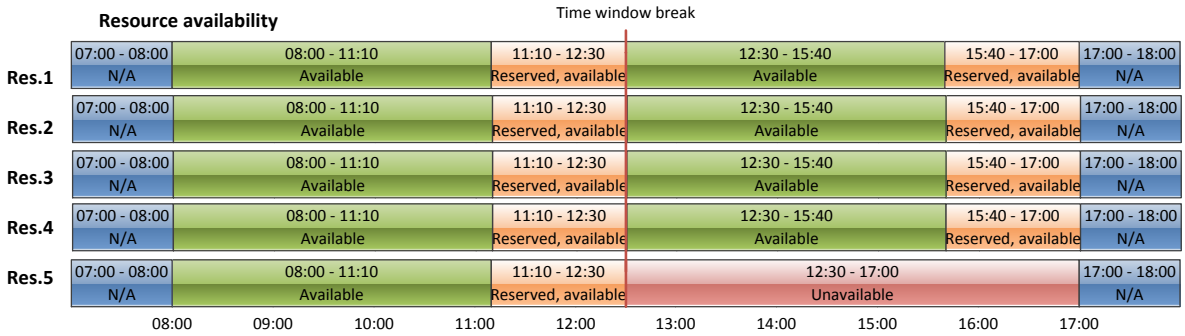


Figure 4.7 Reserved capacity scheduling graph

If decided not to split the planning horizon into multiple timeframes, the reserved capacity is in general clustered at the end of each day, but is skewed over the resources. This does not enable one to deal with the expected realization of uncertainties as the occurrence of ad-hoc events is expected to be spread over the day. While both event and resources have their availabilities that are time dependent, rescheduling in this case is expected to result in infeasibilities (e.g., events cannot be rescheduled to the second timeframe as their availability does not allow this).

A third methodology to schedule reserved capacity is to create dummy ad-hoc events and consider those during scheduling. The times that a dummy event is scheduled are in reality reserved for ad-hoc events. The main disadvantage of this methodology is, is that it is reserved capacity is reserved pure random as the creation of dummy events is done random. This is because one cannot estimate the characteristics of an ad-hoc event.

Dividing the reserved capacity as a ratio over the multiple timeframes, and over all resources, seems therefore most credible and can best be combined with visit related promises to the customer. This way of reserving capacity does not work if timeframes are short and event durations are relatively large. The amount of time windows allowed therefore depends on the duration of events. For example, if events have an expected duration of 2.5 hours, a single resource can perform three of those per day. Working with two time windows splits the reserved capacity over the time windows, but then a time window is not large enough to process any event. In this case it is more logical to have just one time window. As the amount of timeframes increase planning uncertainty (e.g., what event to schedule first in the n^{th} timeframe?), it is wise to limit the amount of timeframes. A maximum of two timeframes is advised when working with a planning horizon of a day.

As scheduling exactly on those times is hard, it is wise to allow some deviation. For example, in each of the two time windows 6.25 hours is reserved over five resources. This equals 1.25 hour of reserved capacity per resource per time window, a reserved capacity of at least 1 hour per resource is allowed. However, the total reserved capacity should (approximately) equal the reserved capacity. So in case resource 1 is scheduled in such a way that only 1 hour of reserved capacity is left, the “missing” 0.25 hours of reserved capacity must be absorbed by the other resources.

If there is little uncertainty related to the occurrence of events, the initial schedule is almost filled completely. In case an ad-hoc event occurs, the initial schedule must be adapted and another event might be deleted from the schedule in order to be able to perform the ad-hoc event. This methodology is

only practiced if there is little uncertainty to the occurrence of events, as it is undesirable for customers and a company to keep on rescheduling each time an event is added to the set of events.

Utilization affects the resource capacity. Human resources for example need time to complete to-dos but no time is scheduled for this. A percentage of time needs to be reserved in order to enable resources to complete those to-dos. This does not require a specific time within a timeframe. The amount of time that must be reserved must be given as input by the system user.

4.2.1.5. Duration of events

The duration of events is assumed to be known. However, as indicated before there is uncertainty related to their duration. Hans et al. (2008) used slack to deal with deviations in event durations. The amount of slack they plan is influenced by a parameter that relates to the probability of events being complete on time. The parameter value is typically based on historical data.

In relation to Newminds' customers it is decided not to schedule slack in addition to the expected durations. Currently only the expected durations are scheduled and in case of major deviations rescheduling is applied. In addition, it is expected that relatively small variations in event durations neutralize each other. A basic duration is assigned to events based on event type. The expected duration of events is based on historical data, and is assumed to give a proper identification of the duration. If the planner expects an event to have a significant different duration, the duration can mutually be adapted. The scheduling engine should be able to recognize major deviations from the offline schedule, and take action in case the remaining part of the schedule is affected by it.

Preparation and completion activity scheduling

Preparation and completion actions are essential for performing the main event. These actions can relate to activities, or to to-dos. To-dos, related to for example documenting, are dealt with by setting a maximum utilization degree on resources. Activities, such as travelling or cleaning, must be scheduled and therefore they must be assigned a duration.

The duration of activities must be determined, and can be best set by estimations. Some activities can directly be estimated in time units; others are primarily measured on another scale and must be transformed into a time scale. However this is not always easy. Consider for example travelling: one knows the current location of both resource and event. Based on the coordinates an estimation can be made about the distance (e.g., straight line distance, or Manhattan distance), in combination with an average speed an estimated duration can be set. Composing a more detailed estimation of the duration is time-consuming and costly as it needs to use Google Maps for this and as all crossings between resources and events must be analysed.

While assigning duration to activities, there are two elements that must be taken into account. First, one should consider whether it is a fixed duration, or a flexible duration. The duration of fixed duration activities is not influenced by the assigned resource and/or the sequence in which events are scheduled. An example is scheduling cleaning for 15 minutes. Activities with a flexible duration are influenced by the assigned resource and/or the sequence in which events are scheduled. An example of this is travelling.

Second, one should consider whether the activity is location bounded. Some activities that are bounded to the event location, and must therefore fit within the customer timeframe. Others are not bounded to the event location, and therefore they do not have to fit within the customer timeframe.

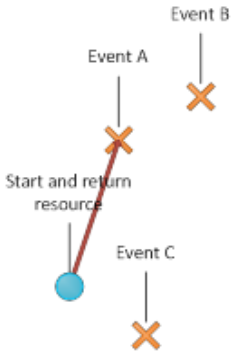


Figure 4.8 Preparation and completion events.

Related to flexible duration activities, the start/return situation must be considered while estimating the duration. The start/return situation represents the state of the resources from where they start and to which they have to return at the end of the scheduling period. For example, resources start at home and have to return there, machines have to be set-up at the beginning and cleaned at the end. The time needed to prepare/complete an event depends on the currents state/location of the resource and its required final state. The required preparation/completion times influence the event selection decision. Let us illustrated with an example (see Figure 4.8) in which three events need to be performed. Event A is scheduled to be dealt with first.

An employee needs to visit events. At the end of the day the employee needs to return home. Event B or event C must be scheduled after event A, only one can be scheduled within the given timeframe. Distance is based on the coordinates of the event (multi-dimension). Based only on the direct distance between events, event B is the closest. Considering the return location, the addition of event C results in a shorter total distance. The decision which event to add therefore depends on the time left within the given timeframe, and the way that weights are assigned to preparation/completion activities, and the consideration of the start/return location.

The influence of the required return duration, on for example the priority value, depends on the structure that is selected. One can decide to fully or partly consider the required return duration.

4.2.2. "When" restrictions

This section describes the "when" restrictions. These are related to specific moments in time, which on their turn relate to allowance of scheduling events. This kind of restrictions can either be a hard or a soft restriction. Hard restrictions must be satisfied; soft restrictions are those that are desired to be satisfied.

4.2.2.1. Working outside resource specific timeframes

Regular working hours are set from 8AM to 5PM. However, the chance that a resource finishes its last event exactly at 5PM is very small. Let us assume that the last event that is scheduled on a resource is finished between 4PM and 5PM. Dependent of the required preparation, event duration and completion another event can be added. Example event A requires 5 minutes preparation, 75 minutes for the main activity and 10 minutes for completion, which is not bounded to the event location. Example event B requires 15 minutes preparation, 60 minutes for the main activity and 10 minutes for completion, which is not bounded to the event location. Let us assume the last scheduled event is finished at 4PM and that both events are available. Figure 4.9 gives the example schedules in case one of the events is added. In both examples the resource is required to work outside of the regular working hours, the ultimate finish time of 5:30PM is met. If none of the example events are scheduled, the resource would be "free" from 4PM; this results in unused resource capacity which is inefficient.

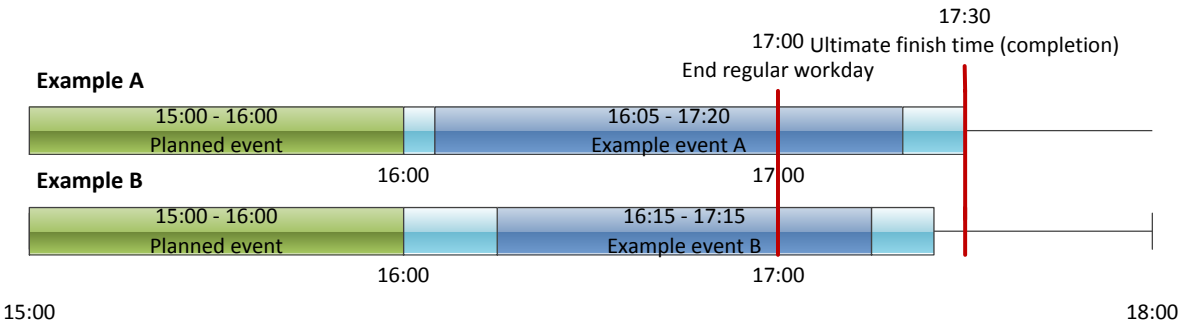


Figure 4.9 Example working overtime

Regular working hours are there for a reason, and so it is clearly undesirable to use resources outside of these time windows. Therefore there should be a balance between the amount of time working outside of working hours (i.e., overtime) and the loss of capacity. The decision whether to add an additional event to a resource, which results in working overtime, is influence by the following factors:

- The amount of time(s) per major time unit (e.g., weeks or months) the specific resource already worked in overtime
- The amount of time working in overtime (and related costs)
  Independent of the amount of time in overtime, the main event must be started before 5PM.
- The unused capacity if no additional event is added
- The option to add another event to the specific resource with less/no overtime
- The costs (i.e., additional value) for not scheduling the event

These factors should be captured in a weight that is considered while calculating the priority value of any combination. At first it is decided to use resource specific timeframes as hard constraints. This implies that working outside those timeframes is prohibited.

4.2.2.2. Working outside customer specific timeframes

Besides the release and due date of an event, the event specific availability must be taken considered. The availability of an event can be influenced by for example the opening hours of the customer. Figure 4.10 graphically represents the availability of an event. Here availability is a simple yes-or-no variable, it is realized that this could be more complex but for simplicity it is assumed to be binary.

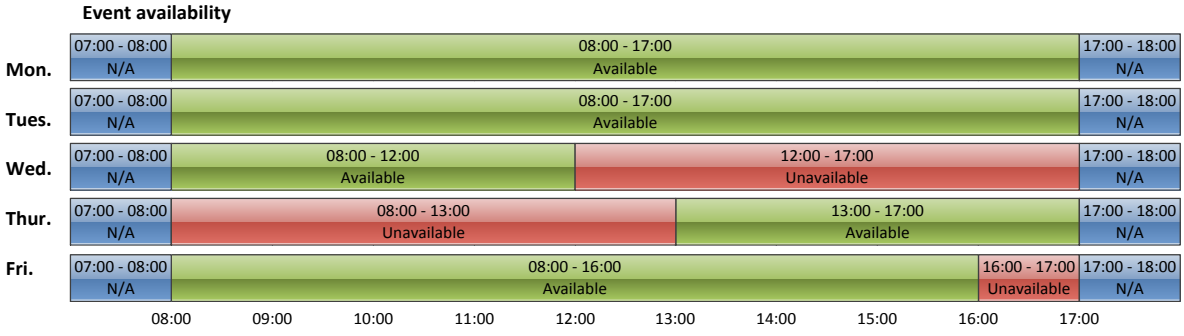


Figure 4.10 Original event availability

As one can see this event is not available at all moments on regular working days. Planning this event based on its availability seems easy, but in combination with all other events it becomes difficult. Let us assume the event is not scheduled on the first two days, but it can be scheduled on Wednesday from 11:15PM to 12:15. Based on the strict availability of the event this would not be an option. However, one must decide whether it is worth moving the event to another time and/or resource. We decided to keep event availabilities as strict deadlines. One exception is made: if the given time window is the last option within the initial timeframe (i.e., initial due date) and the amount of time working outside the customer timeframe is less than half an hour, the event is scheduled at the given timeframe.

Dependent on the type of completion activities, these activities should also be performed within the timeframe. For example, cleaning does have to be performed within the timeframe as it has to be done at the event location, while travelling does not have to be performed within the timeframe as it does not have to be done at the event location.

The availability as given Figure 4.10 is the original event availability. When a specific event is scheduled, there is contact between the executor and the customer in order to update both parties about the event execution plans. As both parties agree on a certain time, it is undesired to reschedule that event in relation to time. From an executor prospective, it can be rescheduled on another resource but the time(frame) the event must be performed is fixed. To make sure these events are not rescheduled, the allowed timeframe is limited to the agreed timeframe. The agreed timeframe must be wider compared to the expected duration, in order to add some slack to the planning and increase the feasibility of fitting other events to the schedule.

This restriction is used as a hard restriction during the design and test phase of the greedy algorithm. This implies that no event can be scheduled outside its given available timeframe.

4.2.2.3. Waiting time

Waiting time for a resource can come in two forms. The first form is expressed in days and reflects on the amount of days an event is already released but not scheduled. This form of waiting time is taken into account when calculating the static priority. Figure 4.11 gives a graphical representation of waiting days. In Figure 4.11 two days are marked as unavailable, these are for example weekend days and are considered as waiting days. One can decide to exclude weekend days (and other holidays) from the waiting time in days.

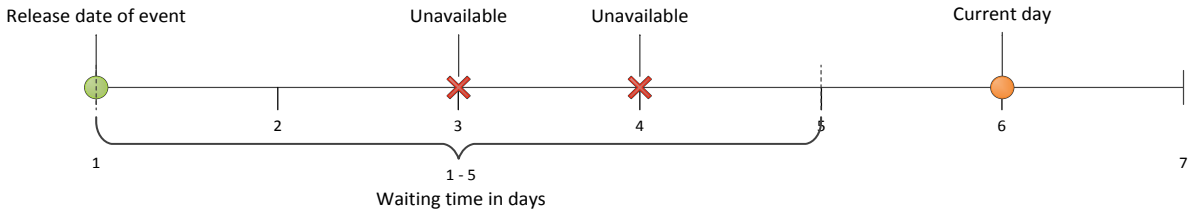


Figure 4.11 Waiting time in days

The second form of waiting time is expressed in hours and relates to the timeframes defined for resources and events. If an event is next to being scheduled on a certain resource, it might be the case that the event is not immediately available. The amount of time between the end time of the

preparation activity and the available start time of the event is referred to as waiting time in hours (see Figure 4.12).

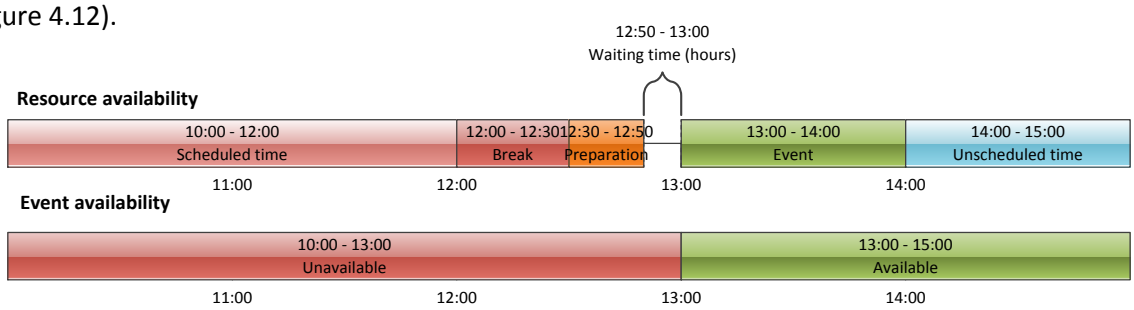


Figure 4.12 Waiting time in hours

This second form of waiting time is considered when matching resources and events. For example, if an event requires two resources, the chance they both arrive at exactly the same time is very small. The difference in time between the arrival of the first resource and the arrival of the second resource is considered as waiting time.

In addition to this, it might be more favourable to schedule another event prior to the event that requires waiting time on the same resource. This in order to fill the gap between the scheduled events, and so the increase in productivity of the resource.

4.3. Summary of scheduling influencing factors

In this chapter the factors that do influence the priority value of an event, or an event-resource combination, are set. The identified factors are:

I. Event type
II. Waiting time of the event
III. Time left till due date
IV. Agreement on timeframe
V. Being a preferred resource
VI. Variable preparation costs
VII. Variable completion costs
VIII. Customer timeframes

There are static factors, which are not relate to time and so the content of these characteristics do not change. For that matter those characteristics are the easiest to consider. The factors that can change over time are harder to merge into the scheduling algorithm, as their content might change over time.

In addition to the priority value, restrictions are identified which limit the options to schedule events. In order to shape those restrictions in GP, some customer input data is required and weights must be assigned. Based on the input data and the weights the scheduling engine is customized. This enables Newminds to provide multiple customers the same GP, while still provide a personalized algorithm.

The following customer resource specific data is required:

I. Number of resources; per resource group and plan group if applicable
II. (Total) resource capacity
III. Availability of each resource
IV. Allowed resource utilization
V. Skills of each resource
VI. Resource steady state / home location

The following customer event specific data is required:

I. Types of event (including expected occurrence rate and duration)
II. Status of events, including identification for allowance for scheduling
III. Required materials / assets
IV. SLA requirements per customer
V. Number of required resources (per resource and plan group if applicable)
VI. Expected duration of main event
VII. Required skills of a resource
VIII. Preferred resource
IX. Availability (including agreed timeframes if applicable)
X. Required machine configurations / location
XI. Required preparation, including location requirement
XII. Required completion, including location requirement
XIII. Ability to outsource

Information about resources and occurred events is assumed to be known within any organization. Information about future events is not directly known within an organization and therefore (historical data) analysis is required. In relation to the realization of uncertainties, the expected amount of new events between offline scheduling and realization of the schedule should be estimated. The other main sources of uncertainty are related to the irregular unavailability of resources and events and deviation in duration of events. In relation to the unavailability of resources there is no measurement that could indicate this. In relation to deviation in duration; the duration of events can best be based on historical analysis. Such analysis should provide one an estimated duration. The estimations must be monitored and adapted in case they do not correspond with reality.

Chapter 5 Scheduling engine

In Chapter 3 we concluded that a dynamic scheduling problem that is applicable in multiple business environments can best be solved by using a constructive heuristic, also referred to as greedy algorithms. Such a constructive heuristic recursively constructs a set of partial schedules from the smallest possible constituent parts. In each step of the greedy algorithm, the best choice of that moment is selected by using a priority rule.

The application of a greedy algorithm can be based on static as well as on dynamic determined priority values. Each time an event can be added to the schedule, the event (i.e., scheduling on static priority values) or the event-resource combination (i.e., scheduling on dynamic priority values) with the highest value is selected for scheduling. Whether it is best to schedule on static or dynamic priority values is not clear. In this chapter we compare various algorithms to compute priority values to schedule the next event. However, the factors considered for scheduling are fixed. In order to test the difference between static priority value scheduling and dynamic priority value scheduling different greedy algorithms are designed. Section 5.1 indicates the steps taken in the different greedy algorithms. Section 5.2 discusses how the different algorithms are tested, and it presents the first noticeable features of those algorithms.

5.1. Scheduling process

This section describes the actions and decisions of different greedy algorithms and how it is applied in the test-environment designed for this research. For each greedy algorithm we present two versions. The first version considers only events that require a single resource. The second version considers events that can require multiple resources (n.b., we consider at most two). Each algorithm is explained by using an example. The example does not consider all identified factors, as the focus is on expressing the different steps and considerations rather than the calculation of priority values.

Algorithm 1 *Scheduling based on static priority values; assigning events to the best resource available, adding new events always to the end of the existing schedule.*

This algorithm schedules events based on their static value. In this case the most valuable event is added to the end of the current schedule of the resource whose combination is most valuable. This implies that the event with the highest static priority value is selected for scheduling. After selecting an event, it is positioned at the first free available moment at the best possible resource. When a second resource is required, the best alternative is selected. **Fout! Verwijzingsbron niet gevonden.** (in Appendix D) displays the flowchart of both versions of Algorithm 1.

The steps taken in this algorithm are visualized in the following example with two resources and four events. The resources are assumed to be available from 8AM to 6PM. The resources have the following characteristics;

Table 5.1 Alg. 1 example resources

Resource	Skills	Location (x,y)
1	1	1, 1
2	1 & 2	10, 10

At the start of the scheduling process, the schedule is empty (see Figure 5.1). The red arrows indicate the moment in time events are possibly scheduled.

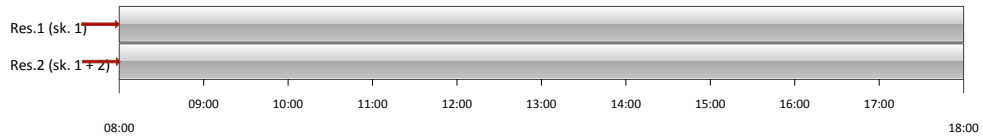


Figure 5.1 Alg. 1 empty schedule

The following events are considered:

Table 5.2 Alg. 1 example events

Event	Static value	Required skills	Duration	Location (x,y)
1	100	1	2h	0, 10
2	150	2	3h	10, 5
3	200	1	4h	5, 5
4	250	2	3h	0, 5

Based on the static value of the event, event 4 is selected to be scheduled first as it has the highest priority value. Based on the required skills only resource 2 is able to perform the event. Event 4 is assigned to resource 2 (see Figure 5.2) at the earliest possible time.

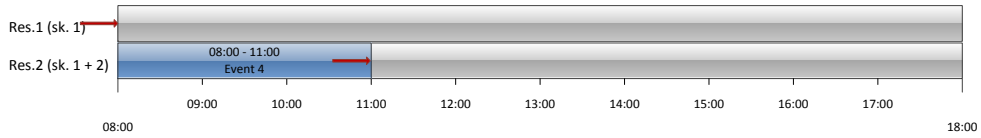


Figure 5.2 Alg. 1 temporary schedule 1

As the static priority value of events does not change during the scheduling cycle, it is not necessary to recalculate the static priority values. So event 3 is next to be scheduled. Based on skills event 3 can be assigned to resource 1 or 2. Considering the travel distances, resource 2 is closer (based on straight line distance: 5 km compared to 5.6 km) and therefore selected (see Figure 5.3).

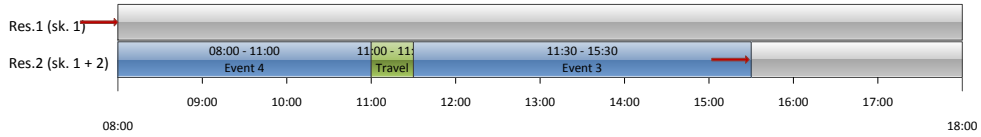


Figure 5.3 Alg. 1 temporary schedule 2

Next in line to be scheduled is event 2. Based on the required skills event 2 can only be scheduled on resource 2, but there is not enough capacity left on that resource and so event 2 cannot be scheduled and is skipped.

Last in line to be scheduled is event 1. Based on the required skills event 1 can be scheduled on resource 1 or 2. The travel distance of resource 1 is based on his current position, which is his home location. The travel distance of resource 2 is also based on his current position, which is at the location of event 3. Considering those travel distances, resource 2 is closer (7.1 km compared to 12.7 km) and therefore selected. Event 1 is added to the earliest possible free moment of the schedule of resource 2 (see Figure 5.4).

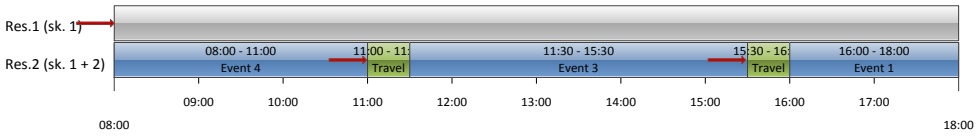


Figure 5.4 Alg. 1 Final schedule example

After selecting an event the travel distance is decisive, the coordinates are therefore important. Figure 5.5 displays the creation of the schedule based on the coordinates. Figure 5.6 shows the final schedule including return routes.

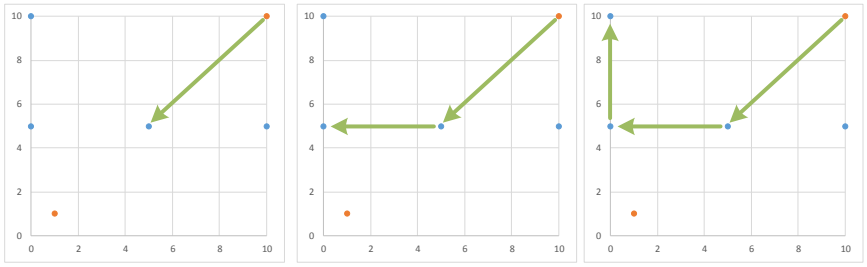


Figure 5.5 Alg. 1 Step by step schedule displayed based on coordinates

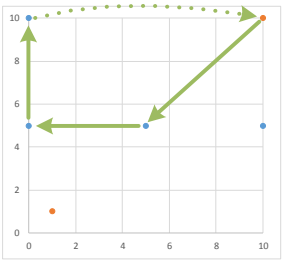


Figure 5.6 Alg. 1 schedule displayed based on coordinates, including return routes

Algorithm 2 *Scheduling based on static priority values; assigning events to the best resource available, adding new events into the best position of the existing schedule.*

This algorithm is similar to Algorithm 1 as they both schedule based on static priority values. In this case most valuable event is added to the best position of the current schedule, considering the movement of other (already scheduled) events. This implies that the sequence in which events are placed is considered. The rest of Algorithm 2 is similar to Algorithm 1. **Fout! Verwijzingsbron niet gevonden.** (in Appendix D) displays the flowchart of both versions of the Algorithm 2.

The steps taken in this algorithm are visualized using the example data used for Algorithm 1. At the start of the scheduling process, the schedule is empty (see Figure 5.7). The red arrows indicate the moment in time events are possibly scheduled.

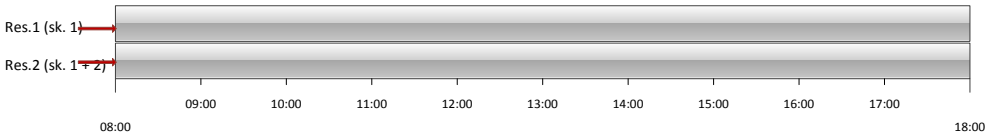


Figure 5.7 Alg. 2 empty schedule

Similar to algorithm 1, event 4 is selected to be scheduled first based on its static priority value. Based on the required skills, only resource 2 is able to perform this event. As shown in figure 5.8, event 4 is added to the earliest possible free moment of the schedule of resource 2.

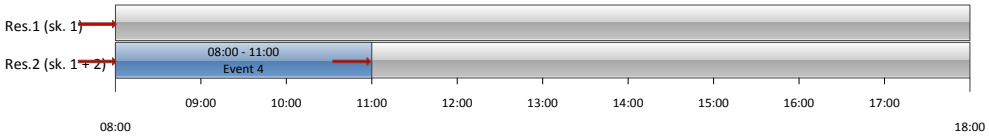


Figure 5.8 Alg. 2 temporary schedule 1

The following event to be scheduled, based on its static priority value, is event 3. As we see in Figure 5.8 there are now three insertion positions for the next event. Event 3 can be performed by both resource 1 and 2, and therefore all thee insertion positions should be evaluated (see Figure 5.9).

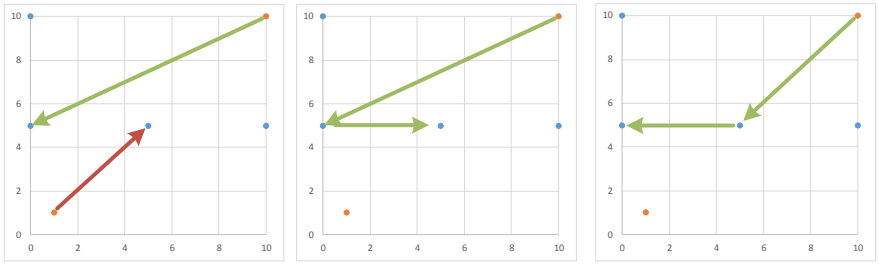


Figure 5.9 Options for scheduling event 3

Option C has the shortest total travel distance, hence event 3 is added to the schedule of resource 2 at the position prior to event 4. In order to create a valid schedule, the algorithm checks the ability to delay the scheduled events. Here we assume that events are available all day, hence event 4 can be delayed without consequences. Figure 5.10 displays the new schedule.

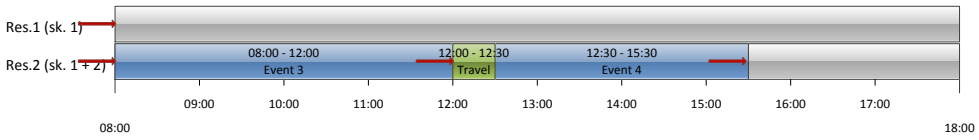
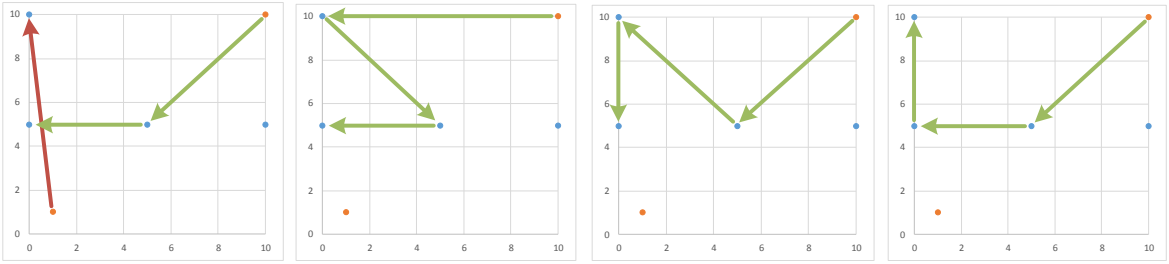


Figure 5.10 Alg. 2 temporary schedule 2

The next event to be scheduled is event 2. Based on the required skills only resource 2 is able to perform the event. However, event 2 requires 3 hours for the main activity. The current end of the

schedule of resource 2 is at 3:30PM and so there is not enough time left to perform this event within regular working hours. Event 2 is therefore not scheduled.

Event 1 is the one to be selected next for scheduling. Based on the required skills the event can be scheduled on both resources. For scheduling event 1 there are 4 insertion positions in the current schedule. Each option should be evaluated.



(a) (b) (c) (d)

Figure 5.11 Options for scheduling event 1

Based on the travel distance option d is most favourable. However, resource 2 is already scheduled until 3:30PM and adding event 1 will cause overtime. Hence this schedule is rejected. Scheduling option A is now the most favourable. As resource 1 is free, this option is selected. Figure 5.12 and Figure 5.13 give the final schedule.

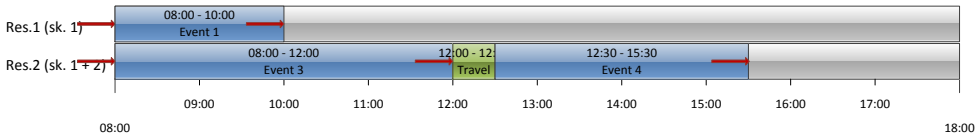


Figure 5.12 Alg. 2 final schedule

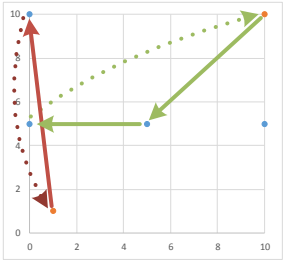


Figure 5.13 Alg. 2 schedule displayed based on coordinates, including return routes

Algorithm 3 *Scheduling events based on dynamic priority values; assigning events to the best resource available, adding new events always to the end of the existing schedule*

This algorithm schedules events based on their dynamic value. This dynamic value includes the distance from the current position of the resource to the event to be scheduled. The most valuable combination is scheduled in each iteration. Events are always added to the end of the current schedule of the selected resource.

An important element in this algorithm is the number of recalculations. Recalculations are required after each iteration, as changed to the schedule result in changed input characteristics. Recalculating can be done in different ways. Option A is to recalculate all possible combinations after each iteration (i.e., adding an event). However, this results in a large number of (re)calculations and can be implemented smarter as the priority value of most combinations remain unchanged. This option is therefore neglected and other options are considered.

Option B creates an array that stores most valuable combination of event and resource for each resource. The overall most valuable combination is added to the schedule. Now only the affected values for a combination of an event and resource are recalculated. That is the best combination of event and resource for the resources that are executing the last scheduled event. We must also recalculate the best event for all other resources that has the last scheduled event as their most valuable.

Option C is to create a list that stores the priority value of all combinations at the beginning. The list is sorted based on the priority value; the most valuable combination at the top, and the least valuable combination at the end. The first combination of the list is added each iteration. Before starting the next iteration, one should recalculate all combinations of the affected resource and again sort the list of priority values. Here one can decide to postpone the recalculation of priority values, as long as no combination that no longer holds (i.e., scheduling on a resources that already has other characteristics due to changes to its schedule) is originally selected for scheduled. However, postponement of recalculations does not significantly reduce the number of calculations, due to the fact that eventually the (re)calculation of priority values must be performed. Only in the last iteration a number of (re)calculations can be saved. In all other iterations one should determine the necessity of recalculating, and so the (re)calculation of effected priority values is postponed. The maximum number of recalculations saved in total equals ((R-1) *(E-1)), where R is the number of resources and E is the number of events.

Comparing the different options shows that option B is the most favourable. This option saves a significant number of calculations and requires relatively small effort. We assume that the events present in the selected combinations for all resources differs in most cases. The amount of recalculations equals the number of events left to be scheduled times the number of resources with the currently scheduled event as part of their best combination. Recalculating the priority value of all combinations (i.e., option A) requires a number of calculations of the order (R * E) each iteration. Recalculating the priority value of only affected combinations (i.e., option B) requires a number of calculations of the order (R).

Sorting all combinations (in option C) requires a number of calculations of the order (E*R log (E*R)). While sorting a list of numbers of length x, there are x! possible outcomes. The number of calculations required to sort an array is of the order x(log x) (Rowell, 2016). The order of calculations is much larger compared to the second option. This all speaks in favour of the second option which is therefore applied in this algorithm.

Summarizing, Algorithm 3 schedules events based on their dynamic value. The most valuable combination is scheduled in each iteration. New events are always added to the end of the current schedule of any resource. For an efficient algorithm we store the best event for each resource and update these values only when this is necessary. At the end of each iteration the priority value of the following resource(s) is recalculated: the resource(s) that is affected by scheduling the event of the current iteration, and the resource(s) whose stored combination holds the same event as scheduled in the last iteration. Fout! Verwijzingsbron niet gevonden.**3** (in Appendix D) displays the flowchart of t his algorithm.

The different steps taken in Algorithm 3 can also be supported by a graphical representation, for that matter the example with two resources and four events is used. For each resource the most valuable combination is stored. This implies that for resource 1 the priority value with event 1 and event 3 must be calculated, and for resource 2 the priority value with all events.

In order to calculate the priority value of a combination both static and dynamic priority values are combination. In the example travel distance is the only dynamic factor. For each km 10 priority value point are distracted. While creating the array the following calculations are performed:

Calculation 1.1: resource 1 – event 1: 100 – (9.1 *10) = 9
➔ Store in array for resource 1
Calculation 1.2: resource 1 – event 3: 200 – (5.6 *10) = 144
➔ Better than stored combination, so replace combination in array for this combination
Calculation 1.3: resource 2 – event 1: 100 – (11.3*10) = -13
➔ Store in array for resource 2
Calculation 1.4: resource 2 – event 2: 150 – (3.6*10) = 114
➔ Better then stored combination, so replace combination in array for this combination
Calculation 1.5: resource 2 – event 3: 200 – (4.2*10) = 158
➔ Better then stored combination, so replace combination in array for this combination
Calculation 1.6 resource 2 – event 4: 250 – (8.5*10) = 165
➔ Better then stored combination, so replace combination in array for this combination

The array contains the combinations of resource 1 with event 3 and resource 2 with event 4. The priority value of the second combination is the highest and therefore selected (see Figure 5.14).

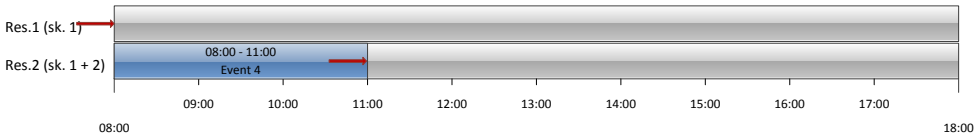


Figure 5.14 Alg. 3 temporary schedule 1

The scheduled event is not in the combination with resource 1 in the array, and so that combinations remains in the array. For resource 2 the combination is removed and so for this resource the new best combination must be found.

Calculation 2.1: resource 2 – event 1: 100 – (5*10) = 50
➔ Store in array for resource 2
Calculation 2.2: resource 2 – event 2: 150 – (10*10) = 50

➔ Equal to stored combination, so no changes in the stored array

Calculation 2.3: resource 2 – event 3: 200 – (5*10) = 150

➔ Better then stored combination, so replace combination in array for this combination

The array now contains two combinations; resource 1 with event 3 and resource 2 with event 2. The dynamic priority value of the combination with resource 2 is the highest (150 compared to 144) and therefore event 3 is added to the schedule of resource 2 (see Figure 5.15).

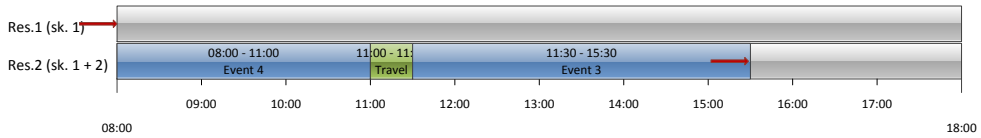


Figure 5.15 Alg. 3 temporary schedule 2

The scheduled event was present on both combinations in the array. Now, for both resources the new best combination must be calculated. Resource 1 can only perform event 1. The priority value of resource 1 and event 1 is 9 (see calculation 1.1). For resource 2 the only option is to combine with event 1, as resource 2 does not have enough capacity left to perform event 2. This implies that event 2 cannot be performed by any resource. The array now contains the combination of resource 1 with event 1 and resource 2 with event 1. Event 1 is therefore added to the schedule of resource 2, following event 4. Figure 5.16 gives the final schedule created by algorithm 3.

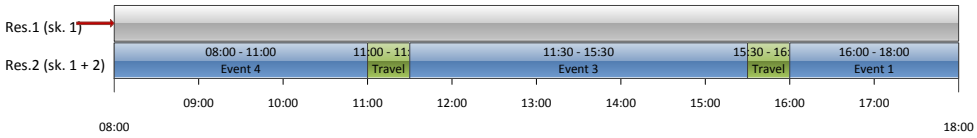


Figure 5.16 Alg. 3 final schedule

Using the example data, the third algorithm gives the same results as the first algorithm. For a schedule based on the coordinates see Figure 5.5 and 5.6.

Algorithm 4 *Scheduling events based on dynamic priority values; assigning events to the best resource available, adding new events into the best position of the existing schedule.*

This algorithm also schedules based on the dynamic value of combinations, but does not require an event to be scheduled at the end of the existing schedule. In this case, the best insertion position for each event is determined. The number of positions is based on the number of events already scheduled (i.e., the number of positions equals the number of scheduled events plus one). Similar to Algorithm 3 we maintain an array with the best event for each resource and update only the values necessary to make this algorithm efficient. In each iteration the priority value for an event on any resource at any insertion position is considered. Companies can restrict certain types of events to be moved to a postponed timeframe, for example one does not want urgent events to be moved forward in time, and so restrictions are set on the event type. **Fout! Verwijzingsbron niet gevonden.** (in Appendix D) displays the flowchart for both versions of the fifth algorithm.

Similar to the other algorithms, the steps taken in this algorithm can also be graphically illustrated. For that matter the example with two resources and four events is used again. The first iteration of the algorithm is similar to the first iteration of the third algorithm. That schedule is therefore taken as a starting point here, however new insertion positions are indicated (see Figure 5.17).

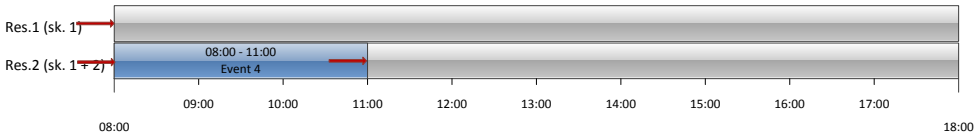


Figure 5.17 Alg. 4 temporary schedule 1

In the stored array there is the combination with resource 1 and event 3. The best combination for resource 2 must be recalculated for each insertion position. When considering the return trip in sequencing two events there is no difference. If there are only two positions at a resource the return trip is not considered, only the additional travel distance. Calculating the travel distance results in the following priority values:

Calculation 2.1: resource 2 – event 1, position 1: 100 – (3.8*10) = 62
➔ Store in array for resource 2
Calculation 2.1: resource 2 – event 1, position 2: 100 – (5.0*10) = 50
➔ Worst compared to stored combination, so no changes in the stored array
Calculation 2.2: resource 2 – event 2, position 1: 150 – (3.8*10) = 112
➔ Better then stored combination, so replace combination in array for this combination
Calculation 2.2: resource 2 – event 2, position 2: 150 – (10*10) = 50
➔ Worst compared to stored combination, so no changes in the stored array
Calculation 2.3: resource 2 – event 3, position 1: 200 – (0.9*10) = 183
➔ Better then stored combination, so replace combination in array for this combination
Calculation 2.3: resource 2 – event 3, positon 2: 200 – (5*10) = 150
➔ Worst compared to stored combination, so no changes in the stored array

The array now contains the combination of resource 1 with event 3 and resource 2 with event 3 on position. The priority value of the first combination is 144 and the priority value of the second combination is 183. The second combination is therefore added to the schedule (see Figure 5.18).

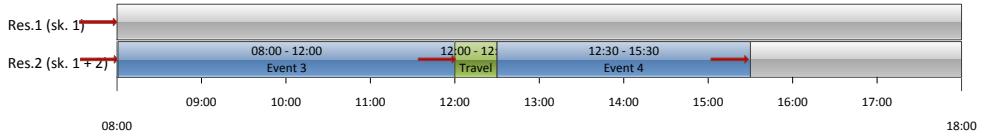


Figure 5.18 Alg.4 temporary schedule 2

After this iteration for both resources the new best combination must be calculated. For resources 1 there is just one option, which is scheduling event 1 with a priority value of 9. For resource 2 there are multiple options that must be considered. However, event 2 cannot be scheduled as there is not enough capacity left.

Calculation 3.1: resource 2 – event 1, position 1: 100 – (10*10) = 0
➔ Store in array for resource 2
Calculation 3.2: resource 2 – event 1, position 2: 100 – (7.1*10) = 29

➔ Better then stored combination, so replace combination in array for this combination

Calculation 3.3: resource 2 – event 1, position 3: 100 – (5*10) = 50

➔ Better then stored combination, so replace combination in array for this combination

Based on these results the combination of resource with event 1 on position 3 is stored in the array. As this is also the highest value in the array, event 1 is added at position three in the schedule of resource 2 (see Figure 5.19).

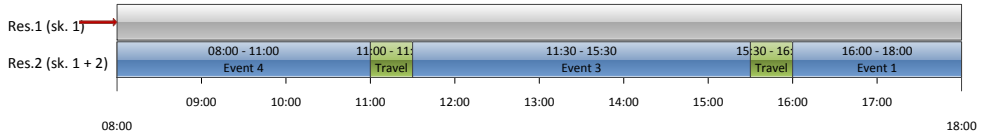


Figure 5.19 Alg. 4 final schedule

Using the example data, the fourth algorithm gives the same results as the first and third algorithm. For a schedule based on the coordinates see Figure 5.5 and 5.6.

The different graphs show the different steps and considerations taken during the execution of each algorithm. Based on the example data the result of the first, third and fourth algorithm are the same, this is not necessarily the case. The overlap in results is due to the simplification and neglect of several identified scheduling factors and the very small data set. In Section 5.2 another (slightly larger) instance is evaluated. The striking features are also discussed there.

5.2. Computational experiment set-up

All designed algorithms are translated into Java-code in order to test the performance of the algorithms. Testing on customer specific data is undesired during the experimental phase as it focusses on a customer specific situation and so the output is customer specific. Therefore, randomly generated input data is used within the computational experiment. The Java-code generates a number of resources with a random location and random assigned qualifications (i.e., skills). Also the set of events is randomly generated and those are assigned a duration, a location, a timeframe (i.e., morning or afternoon), a release and due date, a number of required resources (i.e., 1 or 2) and required skills. The computational experiment gives an indication of the performance of the different algorithms. Locations are used to generate the need for preparation times, as it results in travel distances (or travel times).

In order to create valid and credible experiments Law (2006) indicate the collection of information and data, and the construction of an assumption document as important steps. In relation to the computational experiment data collection is an issue, as any collection is customer specific and therefore the results are not per definition general applicable. For that matter the experimental parameters are based on assumptions (i.e., the parameters and related probability distributions are assumed). For example, the expected duration is assumed to be between 0.5 hour and 2 hours with a normal distribution.

The assumptions made on variables and parameters are based on several data collections from different companies. The number of experiments performed in order to create credible results is set at 50. With this number of runs the influence of outliers due to randomness in the created data is levelled off. In

order to come to this number, several tests are done. Test includes a different number of randomly generated data collections and analyses of the results. A comparison of the average results of the experiment with 5 instances and the experiment with 10 instances is quite large (i.e., over 10 percent). This implies that the performance of those instances is not constant. In order to perform a credible experiment, the results must be more stable and therefore we enlarged the instance size. The difference between experiments with 20 instances and those with 50 instances is small (i.e., lower than 2.5 percent). This percentage does not change when comparing experiments with 50 and 75 instances. As the experiment with 50 instances performs stable, the experiment is performed with this number of instances. The results of the experiment provide insight in the (average) performance of the different algorithms. In order to compare the performance of these algorithm, the following measures are set:

- Total number of events placed
- Time related issues:
    - Percentage of time working on events
    - Percentage of time preparing for events; travel times are used for expressing preparation times. An assumption is made for average speed (i.e., 47km/h). The distance in the algorithms is based on Manhattan-distances.
    - Percentage of time waiting
    - Percentage of time utilized (total time of working, preparing and waiting)
- Sum of static priority value of placed events
- CPU time of the algorithm

The following restrictions are set on scheduling:

- The overall utilization degree is set at 75 percent. This implies that 25 percent of the total resource capacity is reserved to perform currently unknown events. As long as the overall utilization rate is below 75 percent, a new event is added to the schedule. In addition, the time required for return trips are not considered calculating the utilization rate.
- No event can (partly) be scheduled after 5 PM (i.e., overtime is not allowed). This implies that all events must be finished before 5 PM, including return trips.

In order to compare the performance of an algorithm, the results can ideally be compared to the optimal situation. Therefore, a brute force algorithm is created. The brute force algorithm checks all scheduling options given a specific instance and results in the optimal schedule. The running time of the brute force algorithm is very poor, and so it can only be used to test the performance of small instances. The running time of the brute force algorithm takes approximately 3 minutes to solve an instance containing 2 resources and 10 events. The results of the brute force algorithm are compared to the results of the other algorithms. This comparison provides insight in the performance of the designed algorithms, and it indicates the (different) decisions made a certain (critical) points.

As we are dealing with multiple factors, there is no clear definition of an optimal schedule. The optimal schedule is here defined as the schedule for which the sum of the priority values of scheduled events is maximized. Critics can wonder whether maximizing the sum of priority values results in the optimal schedule. Scheduling according to this implies that the duration of an event influences the decision on

what event to schedule as the average added value per time unit increase the popularity of event. Events of the same type (and so the same priority value) can have a different duration. While using the brute force algorithm the short duration event is more favourable than the other as there is more time left to schedule another event. The duration of an event ideally does not influence the priority of an event, but still the goal is to schedule the events that have the highest priority. Within the priority value both the type of event and the urgency are defined. Based on those characteristics the scheduling engine is able to make decisions on what events must be in the schedule. The optimal schedule is therefore defined as the schedule for which the sum of priority value of the scheduled events is maximized.

5.3. Results

Computational experiments are performed in order to review the performance of the algorithms. In this section the results are discussed. The computational experiment is performed in two stages. The first stage is performed in order to gain insight in the decision process, and it is checked for striking features. After making the necessary changes to the algorithms based on those striking features, the second stage is performed. The results of the second stage give the average results of the algorithms as delivered. From the first stage a (relatively small) instance is discussed in more detail. These results provide insight in the decisions made and what can happen, however it can also be an incident. For that matter the different algorithms are tested on a larger scale.

5.3.1. Scheduling instance results and striking features

In Section 5.1 the function of each algorithm is supported by the example instance with 2 resources and 4 events. The calculation of the priority value in that example is simplified, and in addition the instance is small and therefore differences hard occur.

In this section an instance with 3 resources and 8 events is used as an example in which all identified scheduled factors (see Chapter 4) are considered. Each algorithm is applied on the same instance. Figure 5.20 to 5.24 give the resulting routes. This selected instance generates different routes for each algorithm, however this is not always the case. The given results are just an example.

Within the figures the open dots represent the start- and return locations of the different resources. The black shapes represent events. Each event has a number. When an event is in the schedule of a resource there are three numbers displayed next to it: first the number of the visiting resource, second the position and last the event number. The shape indicates the type of event. In this example, the triangles are urgent events, the circles are maintenance events and the boxes are installation events.

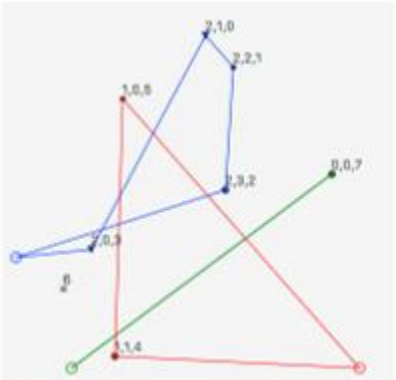


Figure 5.20 Instance results Alg. 1

Figure 5.21 Instance results Alg. 2

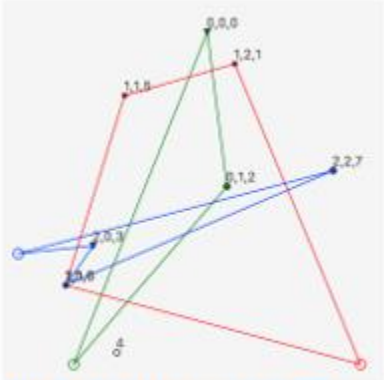


Figure 5.22 Instance results Alg. 3

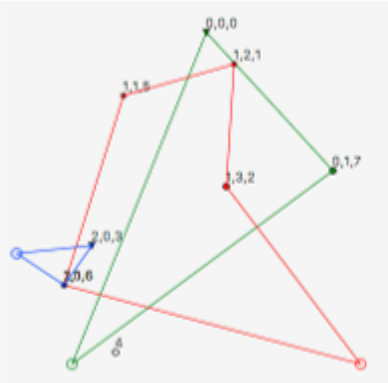


Figure 5.23 Instance results Alg. 4

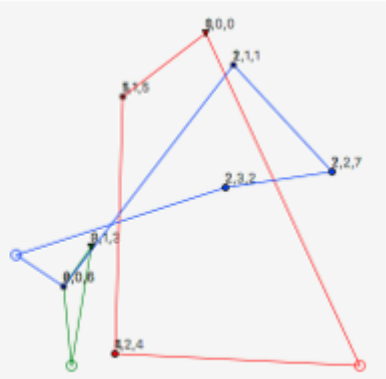


Figure 5.24 Instance results optimal

This particular instance is selected as it provides different routes for each algorithm, which enable us the clearly identify the differences between them an d so striking features can relatively easy be noticed. The striking features are identified and analysed in order to create *face validity* and *result validity*. Face validity is often used in cases where experience and judgement are valued assets. (Shuttleworth, 2009) Face validity measures the degree to which the experiment appears to be reasonable. In addition, we focus on *result validity*. Result validity is concerned with the stability and consistency of the output. In order to create result validity, the negative features initialized by these striking features must be absorbed. The reason for using face and result validity instead of other forms of validity (that are assumed to be 'stronger' or more subjective) is that those forms of validity are general applicable and easily creates feedback.

The found striking features create thoughts on whether one should change the decision making process at some points and how the scheduling process must be further defined in order to perform as desired. The following striking features are found in the preliminary algorithm running results:

1) Not all resources are always used. In the example of algorithm 2 there is a resource that is not assigned any event. Although the decision to not assign any event to this resource is explainable, some customers indicated that it is undesired to pass over resources. At this moment the algorithms provide a schedule that is allowed to pass resource. In the priority value calculation, a weight is assigned to the number events assigned to a resource, but this does not exclude the option to pass a resource.
2) Only the brute force algorithm is able to schedule all events. There is no (significant) difference in the number of events schedule using the other methods. In the example the only difference is in the number of scheduled events that require a second resource. In addition, the algorithms that schedule based on dynamic priority values do provide better results (i.e., schedule a higher sum of priority values) compared to the algorithms that use static priority values.
3) The generated routes do not seem logic as travel distances (i.e., coloured lines in the graphs) are quite long and do cross each other. In general crossing lines in routes are undesired as they indicate as route that is not the shortest. However, when looking in more detail all choices are substantiated.

4) Events are rarely assigned to resources that have bad starting conditions. In the example we see a resource that is less used compared to others, which is partly due to its location. As a result of this we noticed that the vast majority of reserved capacity is allocated at those resources.
5) Scarce skills are used as regular available skills. As a result of this, events that require scarce skills are more likely to not be scheduled as there is no anticipation on the expected use of skills.
6) When multiple events have the same priority value, the decision which event to schedule is often determinative for the whole schedule. In addition, if the priority value of one factor is significant higher than the others, the different between schedules based on static or dynamic values becomes smaller.

In order to respond to those striking features additional weights are considered. The following additional weights are introduced in the algorithms:

1) (Over)classification of resources: *possession of more skills than required by the event negatively influences the priority value of the combination.*
2) Number of events already assigned to a resource: *the more events assigned to a resource, the less favourable the resource becomes. By doing this the likelihood for a resource with a bad starting position to be selected increases. This will spread reserved capacity more evenly over multiple resources. In addition to monitor the utilization on system level, one can limit the utilization of each resource individually (given a certain range). This is not necessary required as assigning weights is easier and (is assumed to) solve the problem.*
3) Weight for multi resource events: *considered in order to prioritize multi resource events. Prioritizing multi resource events will result in a schedule in which (in general) the multi resource events are scheduled prior to single resource events. It was expected that this results in less waiting time, but some test runs do not show a major positive effect. However, this weight is added as scheduling multi resource events is more complex than single resource events, and the proportion of scheduled multi resource event increases using this additional weight. Note, the additional weight is assigned a relatively small value as it should not be decisive for scheduling all multi resource events the first.*

The following additional weight was considered but is not applied in the algorithms:

1) Weight per time unit in relation to the duration of an event: *considered in order to smoothen the average added priority value per time unit of working time spent on the event. However, assigning a weight per unit of working time make event with a long duration more favourable as they will gain more points compared to the same event (of the same event type) with a shorter duration. Considering the average added value per time unit in total, the relative importance of the other factors is diminished as there is no relation time but still the relation is build.*

5.3.2. Scheduling results: general results

The result of scheduling the example instance shows some good results, but it only represents one case. As mentioned before, each algorithm is executed 50 times. For each algorithm the same instances are used within the computational experiment. The experiment is performed for instances of different sizes.

The results are given in percentages, as the relative difference between them is easier to see. Only the CPU is given separately. The following data collection sizes are used and they result in:

Table 5.3 Instance sizes used in the computational experiment

***Instance NR.** **	**1****	**2**	**3**	**4**	**5**	**6**	**7**
# resources	2	2	5	5	50	5	50
# events	10	20	50	500	500	5000	5000

*Instance are referred to as "R... E...". The dots are filled with the number of resources/events
** Optimal scheduling (i.e., brute force) is only applied for this instance size.

The first measure relates to the total number of events placed. In order to compare the performance of the different algorithms over all test instances, the average number of events per resource per algorithm is given in Figure 5.25. The different instance sizes are on the x-axis, the average number of scheduled events is on the y-axis. Table 5.3 gives the overall average number of events per resource per algorithm.

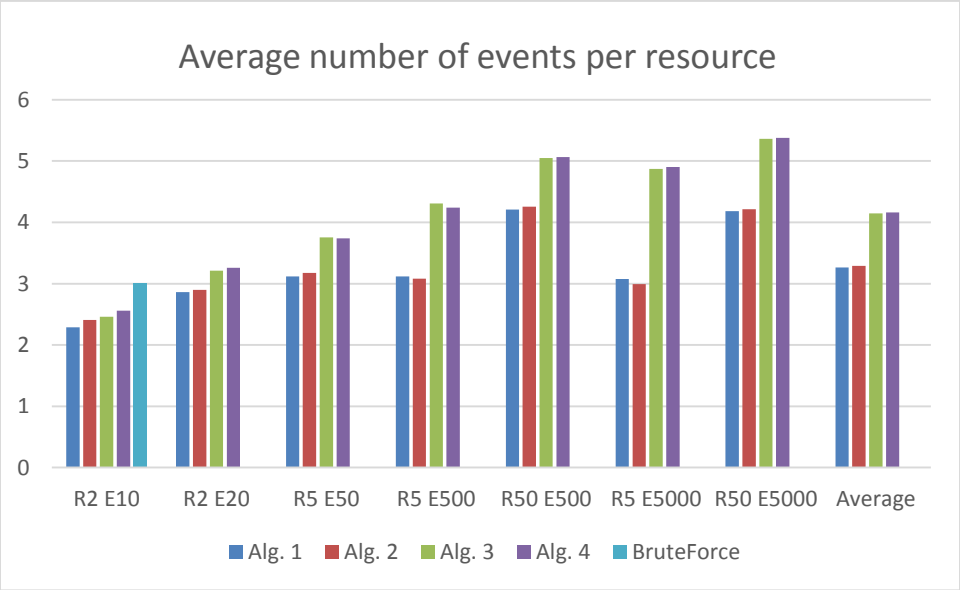


Figure 5.24 Average number of events per resource

Table 5.4 Average number of events per resource and ratio

Algorithm	**1**	**2**	**3**	**4**
Average	3.26	3.29	4.15	4.16
Ratio	78.4%	79.0%	99.6%	100%

Based on Figure 5.25 and Table 5.4 we have a first impression of the performance of the different algorithms. Based on this information we see that algorithms 3 and 4 perform better compared to algorithms 1 and 2, as the average number of events per resource is the higher. This implies that scheduling based on dynamic priority value perform better; on average an algorithm that uses static priority values schedules 20 percent less events.

We also see that the number of available resources influences the number of events scheduled. The instances that both have 50 resources available always schedule more than 4 events per resource, indifferent of using static of dynamic priority values. However, the number of available resources is beyond the influence of the scheduler. Still this is a good indication that the algorithms provide good results on a larger scale, and especially on a larger scale the user will benefit from using the scheduling algorithm.

The increase in the average number of scheduled events with the resources available is explained by the fact that the randomness of resource characteristics is levelled off when having more resources. However, this does not imply that a company should hire as many resources as possible as that will have a negative impact on the utilization of resources. In our experiments there were always more events than could be scheduled during the day considered. Within the pool of resources, a company should strive for overlapping characteristics of resources, as they are able to take over each other's tasks.

The next measures compare the proportional time spent on working, travelling and waiting. The sum of those times give the average utilization rate. Figure 5.26 gives information on the time spent on the different activities. On the x-axis there is the percentage of time spent on a particular activity. The percentage of time is based on 8 available hours per resource per day. Table 5.4 gives the average utilization rate.

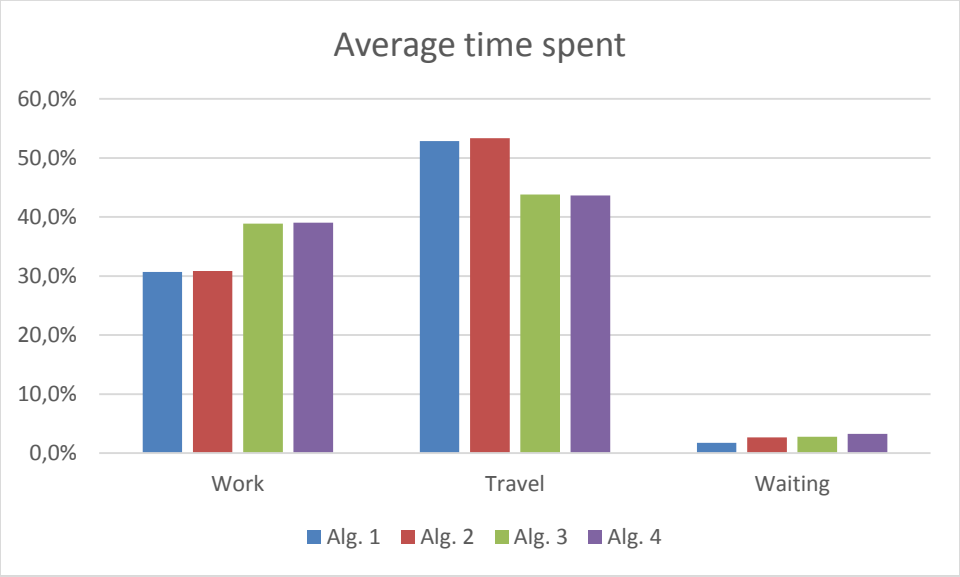


Figure 5.25 Average time spent

Table 5.5 Average utilization

Algorithm	**1**	**2**	**3**	**4**
Utilization	85.3 %	86.8 %	85.4 %	85.9 %

As one can see in Figure 5.26 there is a difference in the way time is spent. Both algorithms that schedule based on static priority values spend relatively much time on travelling, while both algorithms that

schedule based on dynamic priority values spend relatively much time on working. The proportion of time spent on waiting is relatively stable on all algorithms. The increase in waiting time is explained by the increasing number of double resource events scheduled by the various algorithms. When an algorithm does not schedule any double resource events, there is also no waiting time.

Although the way resource spend time differs per algorithm, the utilization rate in all algorithm is approximately 85 percent. This implies that on average each resource has more than 1 hour of free time left within their working hour timeframe. Although a 100 percent utilization rate is hard to achieve, companies strive for a high utilization rate. Having an 85 percent utilization rate here is partly due to the fact that working in overtime is not allowed.

The different between the set 75 percent utilization rate and the achieved utilization rate of 85 percent is due to the fact that return trips are not considered calculating the overall utilization rate. This while the addition of an event will change the time spent on it. Based on this we conclude that 10 percent (i.e., 48 minutes) of the total available time is spend on return trips. However, this depends on the size of the geographical region chosen in the example.

The next measure relates to the sum of priority values scheduled. The achieved priority values are specific for the computational experiment, as the weights of priority values are case specific and therefore instances that use different weights cannot be compared with each other. Within the computational experiment, the priority weights are all the same each experiment. Although the exact weights are case specific, the trends are trustful in general. Figure 5.27 gives the average scheduled priority value per instance size. Table 5.6 compares the different size and generate an overall rate per algorithm.

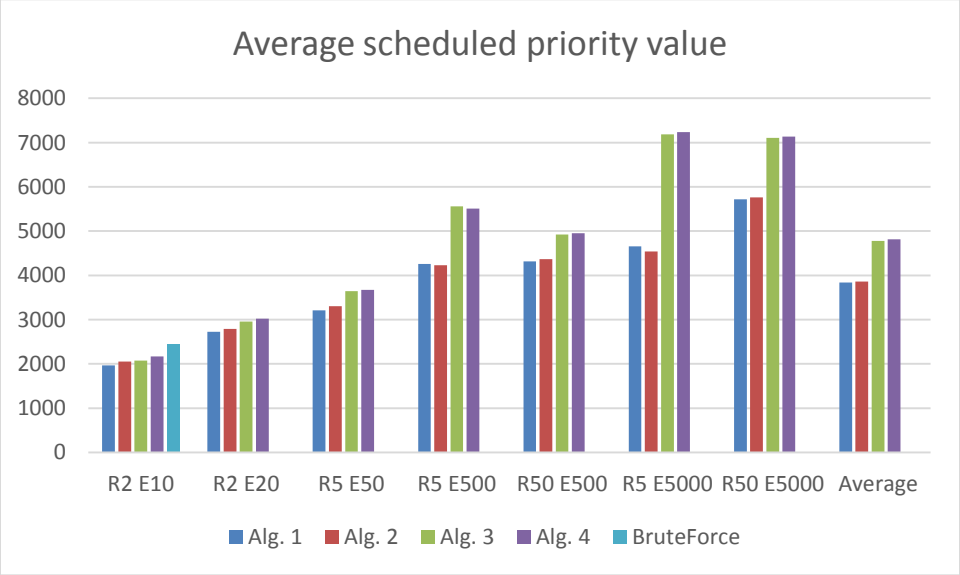


Figure 5.26 Average scheduled priority value

Table 5.6 Average scheduled priority values

Algorithm	**1**	**2**	**3**	**4**
Average	3835	3863	4777	4812
Ratio	79.7%	80.3%	99.3%	100%

Based on Figure 5.27 it is clear that the algorithms that schedule based on dynamic priority values yield schedules with a higher intrinsic priority value. In Table 5.7 we list these numbers in proportion to the average number of scheduled events. Based on those results we conclude that the average value of a scheduled event is the same within each algorithm. However, the algorithms that schedule based on dynamic priority values are able to schedule more event.

Table 5.7 Average event value

Algorithm	**1**	**2**	**3**	**4**
Average event value	1174.9	1174.5	1152.5	1155.9
Ratio	100%	100%	98.1%	98.4%

In addition, we notice that the average scheduled priority value is related to the proportion of events and resources. In general, we can say that the higher the proportion of events, the higher the average scheduled priority value. This can be explained by the fact that a large pool of events increases the chance for a resource to select an event with a relatively high priority value. Two major factors that relate to this are the priority value assigned to event type and the priority value assigned to preparation times. In a large pool of events, there will be more events of an event type with a high priority value. Those event are selected to be scheduled first, and so the sum of scheduled priority values is relatively high. In addition, having more event will increase the probability of having an event close by. The travel distances are therefore smaller. However, based on those numbers we conclude that the weight of the static values is relatively high compared to the dynamic factors.

The last measure is the CPU time of the algorithm. Table 5.8 gives the average CPU time for each algorithm for each different instance size. Based on those CPU times we conclude that only the brute force algorithm is unacceptable, but this we knew before. The CPU time of all other algorithms are acceptable.

Table 5.8 Average CPU times in seconds

	R2 E10	**R2 E20**	**R5 E50**	**R5 E500**	**R50 E500**	**R5 E5000**	**R50 E5000**
Alg. 1	0	0	0	0	0.0054	0.0008	0.0166
Alg. 2	0	0	0	0	0.0108	0.0048	0.0136
Alg. 3	0	0	0	0	0.0100	0.0100	0.1290
Alg. 4	0	0	0	0	0.0312	0.0238	0.3508
Brute force	139.5						

Indifferent of the instance size the results provide a pattern that directs to the best algorithm. The main lessons learned based on the results are:

- Scheduling based on dynamic priority values provide better schedules compared to scheduling based on static priority values. The difference in static or dynamic scheduling is in general unrelated to the size of the instance.
- The difference between the performance of both dynamic scheduling methods are small, but in favour of the fourth algorithm. This implies that sequencing improve a schedule.

5.4. Conclusion

Scheduling events on resources can be done in many different ways. In Chapter 4 the different factors that influence the scheduling decision making process are identified. Based on those factors different scheduling algorithms are proposed in this chapter. The following algorithms are designed:

1) Scheduling based on the static priority value of an event. The events are assigned to the best resource found that is capable of performing the event. The event is always added to the end of the current schedule for that resource.
2) Scheduling based on the static priority value of an event. The event is added at the best (i.e., most valuable) position in the current schedule of any capable resource.
3) Scheduling based on the dynamic priority of an event. Combinations are made between the event and each possible resource. The event is always added to the end of the current schedule of most valuable combination.
4) Scheduling based on the dynamic priority value of an event. Combination are made between the event and each possible resource. The event is added to the best (i.e., most valuable) position in the current schedule of any capable resource.

These algorithms are translated into Java-code and applied on random generated data collections. The results show that scheduling based on dynamic priority values deliver better result, indifferent the size of the test instance. The different between algorithm 3 and 4 are small, but in favour of the fourth algorithm.

Chapter 6 Engine development – further research

The suggested method in Chapter 5 is tested on randomly created data, which sometimes is simplified. In addition, some scheduling factors are simplified or even neglected until now. In this chapter the focus is on these simplifications or neglected issues as they must be considered when putting the algorithm in practice. The simplifications or neglected issues are assigned to one of three sections. Section 6.1 focuses on practical issues related to factors considered in the current algorithm, but which are not fully accountable. Section 6.2 discusses the recommendations and requirements that are necessary before implementing the planning engine. Section 6.3 commences on improvements that can be made on the planning engine in the future.

6.1. Practical issues

Within the Java test environment some issues related to the usage of the algorithm(s) in reality are simplified or neglected. The current considered characteristics are used as key items for making scheduling decisions. The more accurate those decisions are, the better the output (i.e., the schedule) is. In order to improve the scheduling engine the neglected issues should be included and for the simplified issues their effect should be considered. In this section those simplified or neglected issues are discussed one by one.

1) *Reliable estimation of travel distances or travel times.*

   The used travel times and distances are not related to real travel times or travel distances. In the algorithm the travel time is based on the Manhattan distances between locations (i.e., travel distance) and a given speed. To put the algorithm into practice, the estimated travel times or travel distances must be translated into real distances or reliable travel times. For this matter tooling, such as Google Maps, can be used. The main disadvantage of using existing tooling is that it is time consuming to access them, and accessibility is bounded to a maximum number of request. Within the algorithm travel distances or travel times are frequently used, and therefore the use of such tooling is not realistic with these settings.

   As long as the use of tooling cannot easily be merged into the scheduling process as proposed in the algorithm, travel distances or times must be dealt with in another way. A good way to do so is to use estimated travel distance or times. When adding a combination to the schedule, the estimated travel distance or time is checked with the use of tooling. Performing the check will prevent scheduling strange routes (i.e., travelling over water), but does not require the access of the tool to evaluate every combination.

   To estimate travel distances and times the GPS coordinates (i.e., longitude and latitude) of event locations are used. A simple code can calculate the straight line distance between two GPS locations. The straight line distance is smaller then, or maximum equal to, the real travel distance. Research in traffic models have shown that the average difference between straight line distances and their mutually compared real travel distance is 20 percent (Graumans, 2014). A good starting point for estimating travel distances is therefore the calculated straight line distance multiplied by 1.2. In order to translate travel distances into travel times, average speed

can be used. The average speed of commuter traffic is about 50km/h, based on speed trend developments over the last years (Olde Kalter, Bakker, & Jorritsma, 2010)). The average travel distance or travel time can be estimated with the help of these numbers. The output of this must be compared to the results of existing tools. This check should be added to the steps taken by the algorithm. If the estimated travel time consistently deviates from the output of existing tools, the parameters should be adjusted. However, if the estimated travel times only occasional deviates, the parameters are set right and the selected combination should be re-evaluated. If, based on the re-evaluation, another combination outperforms the selected combination, the selected combination should be exchanged for the best combination. Note, traffic conditions (e.g., traffic jam, road closures and rush hours) and their influence on traffic times are not included.

2) *Scheduling in overtime (event specific decisions)*

In the algorithms scheduling an event in overtime is only allowed if the event itself is finished before the given end time (i.e., only completion times are allowed to finish after the given end time). In reality it can be the case that certain events (e.g., emergencies) are allowed to be scheduled outside the regular framework. However, scheduling those events outside the regular time window should only be done if there is no option to schedule them within the regular time window.

The designed algorithm should be applied, but in addition the remaining list of events should be evaluated. Events that exceed a given threshold value (i.e., certain amount of priority value) should individually be evaluated in order to judge their urgency and to make a decision whether to schedule the event in overtime or not. The height of the threshold value depends on the values assigned to the identified scheduling factors and the interest of the company. This process can be automated and added as additional functionality to the designed algorithms.

The application of the additional event evaluation is not yet included in the scheduling algorithms, as they focus on regular scheduling. The current algorithms will schedule most valuable events in the given timeframe, it does not consider working outside the given time window. The additional evaluation is only required if working in overtime is (partly) allowed.

3) *Filling to the maximum capacity*

The current scheduling algorithms allow filling the overall schedule until a given utilization rate. Filling the schedule until the given utilization rate leaves space for ad-hoc occurring events. The height of the utilization rate (i.e., the amount of time that can be scheduled which equals the total available time minus a reserved amount of time) differs per company and can best be determined based on historical data analysis. Historical data analysis shows the average amount of time the company spends on ad-hoc events per time unit.

The designed algorithms use an overall utilization rate, as it is more useful compared to the use of dedicated reserved resources (see Section 3.4). However, the study of Wullink et al. (2007)

does not consider the proportional division of utilized time over the different resources. This still can result is a division such that all reserved capacity is allocated at certain resources (i.e., it is unlikely events are scheduled at unfavourable resources which will result in a low utilization rate of those resources).

In addition to the current utilization rate, the resource individual utilization rate should be monitored. In order to make sure all resources are used, the restriction that all available resources should have at least one event scheduled can also be added. The current algorithms favours resources that are less utilized, however no penalties are applied for not using a particular resource. The minimum utilization rate of individual resources should be set less than the overall utilization rate. For example, if the overall utilization rate is set at 85 percent, the minimal individual resource utilization rate is set at for example 60 percent. Based on the individual utilization rate, less utilized resources are favoured to be used. Setting an individual minimum utilization rate makes sure all resources are used, but at the same time it allows some deviation between individual resources.

4) *Scheduling over multiple timeframes*

The current designed algorithms create a schedule that covers one scheduling horizon, that is currently set on one working day (i.e., from 8Am till 5 PM). The length of the scheduling horizon can be adapted to the desire of the user. The algorithms consider a scheduling horizon in which resources are uninterrupted available from the start time till the end time. However, multiple companies are working with multiple timeframes within one scheduling horizon. For example, on a working day schedulers communicate to a customer that they will visit in the afternoon. In this case the scheduling horizon (i.e., one day) is divided into two parts (i.e., morning and afternoon).

Algorithms 2 and 4 are able to directly deal with those decisions. In such a case the availability of the event must be adapted. As the scheduler made a promise, the weight of that particular event must be relatively high as the event becomes likely to be scheduled. This weight is currently not specifically mentioned. However, by changing the availability of an event the remaining time to schedule the event becomes smaller and so, based on the assigned weights, the priority value of the event will increase.

As algorithms 2 and 4 are considering different positions to place new events, adding more events within the current sequence of events is possible. The flexible position, and so the flexible timeframe to schedule events, allows movements.

Within algorithm 1 and 3 it is harder to schedule events for which timeframe agreements are made with a customer. In these cases, scheduling events in the afternoon will create a gap that is not filled by the algorithm. Additional functionalities must be designed in order to fill the unscheduled times. As Algorithm 4 is able to deal with timeframe requirements and it outperforms all other algorithms, designing such additional functionalities is unnecessary.

If one wants to schedule for a longer period (i.e., multiple scheduling horizons at once), an additional function must be designed that repeats the original algorithm multiple times. However, additional to just repeating the algorithm, it might be good to investigate the intermediate relations between the different scheduling horizons. An additional element that for example can be considered, it a forecast of expected bustle. Based on expected bustle in upcoming scheduling horizons, it can be good to schedule work in overtime on a prior scheduling horizon. These relations are not yet considered.

5) *Worktime regulations are not considered*

The current algorithms do not consider any worktime regulation, such as lunchtime or maximum hours of intermittent work. In deliberation with the user these regulations should be mapped and translated into scheduling constraints.

An easy solution method can be given for scheduling lunch breaks: the event that crosses the timeline of 12AM should be extended by half an hour. By doing so the lunchtime is not really scheduled, but the field engineer has half an hour of time for lunch. The field engineer can determine for itself whether to have lunch during the performance of the event or to have lunch prior to starting or after finishing the event.

Dealing with maximum hours of intermittent work and different types of shift is more complex. There are all kind of rules that limit the deployment of resources. For human resources in the Netherlands those regulations are set in the *arbeidstijdenwet*. Those regulations are another level of scheduling and are therefore out of scope. However, they influence the availability of resources which must be known for operational scheduling. The output of those regulations should therefore be considered.

6) Outsourcing of events

The designed algorithms consider known and available resources. However, some companies have the option to hire extra resources in order to enlarge their resource capacity. The decision whether to hire extra resource capacity is often made based on the expected bustle. In relation the scheduling engine, the decision whether to enlarge the resource capacity can be made based on the total duration of events that must be scheduled within the upcoming scheduling horizon(s). In order to decide whether to hire extra capacity or not, the urgency of the events and the total duration of events are important. For example, if there are 100 events (each with a duration of 1 hour) on the list to be scheduled and 20 of them must be performed before the end of the next scheduling horizon. With only two resources available (i.e., 16 available hours) there is not enough capacity to perform all events that must be performed. If possible, the company should hire extra capacity because they know beforehand that it is impossible to schedule all events that officially must be done. The scheduling engine should determine when extra capacity is desired and give a signal to its user.

6.2. Implementation

The designed algorithms are tested by using random generated test data. However, before applying them in reality some implementation recommendations and requirements are given. In this section those recommendations and requirements are discussed. The key issues relate to face validation of the tested scheduling engine (Section 6.2.1.) and the settings of the priority values (Section 6.2.2.).

6.2.1. Validation and benchmarking

Although the algorithm and their result are tested for face validity (see Section 5.3.1.), actual face validity must be created during the implementation phase of the automated scheduling tool. Face validity is a result of testing and analysing data gathered under real life circumstances. As a part of the development phase, the scheduling engine must be put in practice and feedback from its first user should be gathered. Gathering and processing the feedback is an iterative loop during the development phase. Once the performance of the scheduling engine is satisfying, the scheduling engine can be prepared for real implementation. However the feedback loop never ends, as changes in the process or branch may occur. In order to deliver good results (i.e., useful schedules) the scheduling engine should keep up with customer and market developments. Figure 6.1 gives the different phases in the development of new systems. This research gives direction to the design. Before the scheduling engine is ready to be developed, face validity must be created for the application of the scheduling engine. This must be done for each individual company.

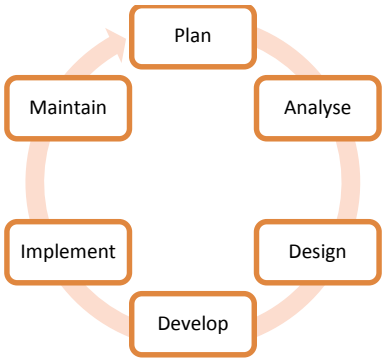


Figure 6.1 Systems development life cycle (Sparkdawn)

In addition to (individual) face validity benchmarking is recommended. Benchmarking is a systematic comparison of (organizational) processes and performances within the same niche or branch. The general process and their belonging characteristics are discussed in Chapter 2. However, the demarcation made there should also be verified. Verification is concerned with the quality of the provided solution (i.e., does the solution meets the specifications?), validation is concerned with the usefulness of the provided solution (i.e., does the specification meet the customer needs?). (de Graaf, 2014) Together verification and validation are used to demonstrate that the given solution meets the requirements and needs of the customers, and thus the solution fits within the solution space.

Benchmarking focuses on verification issues, and therefore the focus is not primarily on collecting and analysing data, rather the focus is on the thoughts behind the system. The thought behind the system are likely to be similar within the same niche, as the same key characteristics are involved. Reliability and

representativeness are therefore two important elements in benchmarking. However, in order to compare individuals within the same niche one should be aware to compare the same issues. Therefore definitions of what is measured/analysed should be clearly defined. Newminds' position as a solution provider to multiple customers within the same branch is a good starting point for initializing benchmarking. From their position they are able to compare the process and belonging main characteristics of the scheduling process as practiced within a certain niche of branch.

6.2.2. Sensitivity analysis on priority values

While testing the algorithms on random data, the priority values assigned to the distinct factors are similar during each run because otherwise the results are non-comparable. However, the values of those factors are set based on instinct rather that they are well substantiated. The initial priority values are adapted while testing in order to create face validity, however customers might require different setting.

The priority value assigned to the distinct factors will differ per company, dependent on their priorities and the interrelations between the different factors. Some factors relate to one another, therefore their value is also important. For example, the value of a preferred resource depends on others, as the preference of one resource does not say that resource must be assigned to that particular event. The consideration whether to assign the preferred resource depends for example on the required additional travel distance. If the preferred resource is close (e.g., less than 20 km additional travel distance compared to other resources) one would assign the preferred resource. In this case, the priority value assigned to the preference of resource therefore depends on the priority value assigned to travel distance (per travel distance unit).

The dependencies between those factors are company dependent. Each company that uses the scheduling engine should set their own priority values and validate the scheduling engine performance. However, they should be aware of the fact that one factor can overrule the entire system. This is the case when the different priority values are not balanced. In that case the system becomes more sensitive for the appearance of certain characteristics than others. In addition to verification (i.e., benchmarking) and validation also sensitivity analysis on the priority values is recommended.

Setting the priority values within a company can initially best be based on a combination of gut feeling and experience. However, many companies have similar settings. For example, almost any company will prioritize urgent events. Therefore the priority value of such events must be higher compared to others. Which characteristics to favour are often known within the company, and so they can initially be rated. With the initial settings the company should test the automated GP for face validity. The priority values create a priority rule that is specific for each company. Each individual company should be aware that the set priority values are hard restrictions, and so all decisions are made on those values. Although the algorithms can be expanded with many characteristics, it is advised to limit the number of characteristics at first. While putting the automated GP in practice, the priority values can recurrent be adjusted and missing characteristics can be added.

6.3. Improvements

The designed scheduling engine captures most important scheduling characteristics. However, the more accurate the input and scheduling decisions are made the better the output. In order to improve the scheduling engine there are two options, which are not mutually exclusive. The first option is to improve the accuracy of the current considered characteristics, the second option is to include more decisive characteristics.

The accuracy of the current considered characteristics is already discussed in Section 6.1. This section commences on those characteristics that are not yet included as they are not primarily decisive in scheduling, however they can improve scheduling process results when considered.

Relate to the expansion of included scheduling factors, one should always remember that the system captures a process that is subjected to personal preferences. In manual scheduling system, planners constantly make trade-offs which are based on a lot of characteristics, but many decisions are influenced by the person who schedules. The scheduling engine capture at least most influencing factors, but can be further expanded by a lot of factors. Together all those factors should simulate the decision making process of the planner. A down side on this is that the more characteristics included, the more difficult it gets to create a schedule. Each company should make a trade-off between the factors they want to include, such that the system provides acceptable results and is controllable.

The decisions made in the current scheduling algorithms are all local decisions. Each decision is made based on a timeframe, a list of events, and a list of resources. To narrow the view, the more chance to deviate from the path to optimality as interrelations are not considered. If the effect of scheduling a particular event, resource, or timeframe (and thereby removing that option from the list of options to be scheduled in the future) can be known beforehand, scheduling can be done more accurate as the effects can also be considered. For this matter a 'look ahead' functionality can be useful.

Examples for factors to consider as part of the 'look ahead' functionality are:

- *Time prior to the opening of a timeframe*
  Timeframes that are not open at the moment of scheduling are currently not considered for that scheduling decision. However timeframes that are about to open might be valuable option for scheduling. This can especially be the case in preparation times are involved.

  For example, event A is available from 12AM to 5 PM and event B is available from 8AM to 14PM. The current time for scheduling is 11AM. According to the availability timeframe of events event A cannot be scheduled yet. However, travelling to event B might take more time than travelling to event A and waiting there until it is 12AM (i.e., the preparation time for event A equals at least 1 hour). Therefore decisions can better be made based on total preparation times (i.e., regular preparation time plus optional additional waiting time until timeframe opens) for each event.

- *Time left within the given timeframe*
  In the current algorithms the number of days left to schedule the event are considered. However, the time within the current (daily) timeframe is not considered. As an extension also the time left within the current timeframe and possible effect of not selecting the current time for scheduling can be considered. For this the same further reasoning can be applied as above.

In additional to these improvement suggestions, also the following improvement can be considered:

- *Method for selecting a second resource*
  The current algorithm selects, if necessary, a second resource based on the 'best alternative' principle, in which the event and time first resource are already fixed. However, if that event is best to be scheduled another couple of resources might be a better option, although they are not the overall best option for that particular scheduling iteration. In order to improve the schedule engine, considering a resource couple while evaluating the priority value of a combination is valuable.

6.3.1. Further research

Within this research multiple solution approaches are discussed and evaluated, and based on our findings and the experiment results the application of Algorithm 4 provides a solid foundation for a scheduling engine. Gere (1966) states that the selected solution approach should include an answer to the question. As Algorithm 4 provides an answer to the question, the selected approach is useful. However, in addition we should ask ourselves what should be done, in case the result (i.e., schedule) is not good enough or if we think we can do better. The CPU times of Algorithm 4 within the test environment are much smaller than expected beforehand. Applying Algorithm 4 within a real life functioning system, considering real life data and all data sources, the CPU time might increase but still it is expected to be within acceptable boundaries. Due to the fact that the CPU times of Algorithm 4 are much smaller than expected beforehand, it is also worth considering applying an improvement method in addition to the current algorithm.

Likewise the wide range of constructive methods, there is a wide range of improvement heuristics. Simple improvement heuristics are for example r-opt. R-opt swaps r events and check whether the new solution is an improvement. If so, the new solution is accepted. If not, the original solution is kept. However, over the last years the application of metaheuristics as improvement method have increased popularity. The concept of metaheuristics is discussed in Section 3.3.2. Based on the earlier given descriptions as small desk study is performed into the successful application of metaheuristics as improvement methods, especially in scheduling. As a result of this search I recommend further research into the application of Tabu search.

Tabu search was first introduced by Glover around 1980. Over the years Glover successfully applied Tabu search on multiple scheduling problems. Through the development of Tabu search over the years, it has proven to find superior solutions and it has demonstrated to have advantages in the ease of implementation and in the ability to handle additional constraints (i.e., flexibility).

As mentioned in Section 3.3.2. Tabu search is a deterministic local search strategy that is able to escape from local optima. The schedule generated by Algorithm 4 can easily be used as a starting point for Tabu search. In order to improve the current solution, events are swapped and the new (neighbourhood) solution is stored on the Tabu list. The Tabu list prevents short-term cycling. The application of Tabu search increases the required running time, but is also expected to increase the quality of the results.

6.4. Conclusion

The currently designed algorithms do consider most important scheduling factors, however before implementing the scheduling engine some actions must be taken. If the scheduling engine is developed, also some improvements can be achieved. This chapter was devoted to mark the actions and considerations required for development, implementation and improvement of the scheduling engine.

Within the used algorithms and test environment some issues are simplified or neglected. Although the current considered characteristics are the key characteristics in scheduling decision making, actions and considerations related to the development of the scheduling engine are given. Those actions and recommendations relate to:

1) Reliable estimation of travel distances or travel times.
2) Scheduling in overtime (event specific decisions)
3) Filling to the maximum capacity
4) Scheduling for multiple timeframes
5) Worktime regulations are not considered
6) Outsourcing of events

Besides those simplified or neglected factors within the test environment, one should be aware that the algorithms are currently only tested within a simulated environment. Before putting those algorithms in practice the scheduling engine should be tested on real customer data. For that matter the following actions should be taken:

- Creating face validity
- Benchmarking branch/niche processes and performances
- Sensitivity on priority values

After above mentioned actions, and the actual development of the scheduling engine, the scheduling engine can be put in practice. However, the scheduling engine can further be developed by improving the accuracy of the input values, or improving the quality of scheduling decisions. For that matter a ‘look ahead’ functionality is useful. In addition, the methodology for selecting a second resource can further be improved.

Chapter 7 Conclusions

This chapter summarizes the research conducted at Newminds make IT happen B.V. The central research question will be answered. To this end all relevant actions, demarcations, discoveries and choices are presented.

The research conducted is divided into several elements which encompass the organizational environment and the product central to this research, a generalisation of the application window of the product, existing literature concerning resolving the research problem and possible solutions. By means of this elements an answer to the below mentioned research question is provided.

"Whilst enhancing the existing scheduling tool, what solution method to automate the scheduling process is optimal for Newminds, when multiple scheduling influencing factors are taken into consideration?"

The overall objective of this research is to create a functional model, the "scheduling engine". Its desired functionality includes multiple factors and should provide GP users with an automatically generated schedule that considers the identified factors.

The scheduling tool

As indicated by the first part of the research question, there is an existing scheduling tool that should be extended. In order to do so, the current state of the scheduling tool must be surveyed. The current scheduling tool supports planners to keep track of the scheduling process and to make changes to the current schedules based on transformations in the dataset. Manual actions of a planner are required for (re)scheduling. To ensure a feasible schedule, the planner must take into account multiple factors. At the moment of writing, Newminds is expanding the system options and adding certain features, to ensure an increase in planner's insight to the feasibility of scheduling options. However, these expansions do not automate the scheduling process itself.

In order to design an automated scheduling system, which is desired to be broadly applicable, the fundamental decision-making process for scheduling in organizations must be mapped. The scheduling function may differ amongst organizations, for example based on the product or service they deliver, but one may also expect similarities. The similar decision factors should serve as basic functions for automated scheduling.

Constraints for scheduling engine development

In regards to the design of the automated scheduling tool there are some constraints. The most significant limitation is the requirement that states that the automated scheduling tool should be implemented within the current information technology infrastructure. This denotes that no current relations between software applications or servers are changed, and that no additional servers should be required. The only addition that is permitted is the processing-element that deals with the scheduling process (i.e., scheduling engine).

Additionally, the automated generated schedule should represent the current way of scheduling. Furthermore, the tool should generate a feasible schedule within reasonable time. In order to achieve this possibilities and limitations in regard to GP's data infrastructure and GP's data processes are

considered. As the generated schedule must be realistic at any time, the use of real-life information is required. Nevertheless, offline gathered data can be used to build the schedule initially. The reasonable amount of time for generating a schedule depends on the strategy that is used: a schedule with little related uncertainty, and so less expected (re)scheduling actions might take more time compared to a schedule that requires more rescheduling actions. As (re)scheduling is done at the operational level, and decisions must be made quickly, a schedule must be given (almost) instantaneously.

General inputs for scheduling

The scheduling process always involves a set of resources and a set of events. Resources are a type of unit that are able to transform some type of input into a different output. Resources are characterised by their availability, capacity, utilization, skills and location. Events represent the jobs/orders that require action in order to be performed. Events are characterised by their type (and required skills), status, processing stage, release and due date, location and allowance of pre-emption.

In order to create a schedule, resources and events should be assigned to each other. This relation is based on the required characteristics on the event side, and the deliverable characteristics on the resource side. However, matching those characteristics is complex as decisions are not univocal.

The high amount of characteristics and factors that influence the scheduling process makes it challenging to create a scheduling algorithm that considers all factors and their (mutual dependent) relations. Therefore, the above given demarcation of characteristics and factors is constructed.

Solution directives

Scheduling problems are very dynamic and can, due to their size, hardly be solved to optimality. For that matter the focus shifts to approximation solution methods, also referred to as heuristic. Evaluation criteria are set to discover a suitable solution method. Based on the evaluation criteria regarding accuracy, speed, simplicity and flexibility, different solution methodologies are analysed. The accuracy and speed will be determined by the selected heuristic. The simplicity of basic heuristics supports acceptance of the solution. Flexibility is a must, as the solution must be general applicable and accommodating different customers.

However, no heuristic is found that fits to the research problem. Therefore a customized greedy algorithm seems the most logical option. A greedy algorithm works by recursively constructing a set of objects out of the smallest possible constituent parts. The following algorithms are designed:

1) Scheduling based on the static priority value of an event. The event is added to the end of the current schedule for the best resource found that is capable of performing the event.
2) Scheduling based on the static priority value of an event. The event is added at the best (i.e., most valuable) position in the current schedule of any capable resource.
3) Scheduling based on the dynamic priority of an event. Combinations are made between an event and each capable resource. An event can only be added to the end of the current schedule of a resource.

4) Scheduling based on the dynamic priority value of an event. Combinations are made between an event and each capable resource. An event is added to the best (i.e., most valuable) position in the current schedule of any capable resource.

These four algorithms are translated into Java-code and applied on random generated data collections. The accompanying results show that scheduling based on dynamic priority values delivers the best result, indifferent the size of the instances. The difference between algorithms 3 and 4 is small, but they are in favour of the Algorithm 4. Therefore, it is recommended to use of the Algorithm 4.

Development

The currently designed algorithm do consider most important scheduling factors, however before implementing the scheduling engine some actions must be performed. If the scheduling engine is developed, also some improvements can be achieved

While testing the algorithm some issues are simplified or neglected. Although the current considered characteristics are the key characteristics in scheduling decision making, the following improvements of considered factors are recommended: reliable estimation of travel distances or travel times, scheduling in overtime (event specific decisions), filling to the maximum capacity, scheduling for multiple timeframes, worktime regulations are not considered and outsourcing of events.

Besides those simplified or neglected factors within the test environment, one should be aware that the algorithms are currently only tested within a simulated environment. Before putting the algorithm in practice the scheduling engine should be tested on real customer data. For that matter the following actions should be taken:

- Creating face validity
- Benchmarking branch/niche processes and performances
- Sensitivity on priority values

After above mentioned actions, and the actual development of the scheduling engine, the scheduling engine can be put in practice. However, the scheduling engine can further be developed by improving the accuracy of the input values, or improving the quality of scheduling decisions. For that matter a ‘look ahead’ functionality is useful. Also the methodology for selecting a second resource can further be improved.

Final answer

Given the above information, the main research question can be answered. The best solution method to be used by Newminds, in order to automate the scheduling process within the existing scheduling tool, is to develop a greedy algorithm that assigns combinations of events and resources to a specific time based on dynamic priority values.

Bibliography

Bard, J., & Purnomo, H. (2005). Preferense scheduling for nurses using column generation. *European journal of operational research* , 510 - 534, Vol. 164.

Billaut, J., Moukrim, A., & Sanlaville, E. (2008). *Flexibility and robustness in scheduling.* London: ISTE Ltd (in coorporation with John Wiley & Sons, Inc.).

Cook, W. (2008, March). *Optimal 85,900-City Tour*. Retrieved May 2016, from TSP - The traveling salesman problem: http://www.math.uwaterloo.ca/tsp/pla85900/index.html

Cordeau, J., Gendreau, M., Laporte, G., Potvin, J., & Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the operational research society* , 512 - 522, Vol. 53.

Farlax Inc. (n.d.). *Schedule definition*. Retrieved May 28, 2015, from The free dictionary: http://www.thefreedictionary.com/schedule

Geffner, H., & Bonet, B. (2013). *A concise introduction to models and methods for automated planning.* San Rafael: Morgan & Claypool publishers.

Gendreau, M. (2002, September 05). Metaheuristics in action: lessons learned from implementations in vehicle routing . Olso, Norway.

Gendreau, M., & Potvin, J. (2005). Metaheuristics in combinatorial optimization. *Annals of operations research* , 189 - 213, Vol. 140, No. 1.

Gere, W. (1966). Heuristics in job shop scheduling. *Management Sci.* , 167 - 190, Vol. 13.

Ghallab, M., Nau, D., & Traverso, P. (2004). *Automated planning; theory and practice.* San Francisco: Morgan Kaufmann publishers.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and operations* , 533 - 549, Vol. 13.

Graumans, E. (2014, 11 24). (in Dutch) Het beste verkeersmodel is de werkelijkheid!

Hans, E., Wullink, G., Van Houdenhoven, M., & Kazemier, G. (2008). Robust surgery loading. *European journal of operational research* , 1038 - 1050, Vol. 185.

Haugen, D., & Hill, A. (1999). Scheduling to improve field service quality. *Decision Sciences* , 783 - 804, Vol. 30, No. 3.

Haupt, R. (1989). A survey of priority rule-based scheduling. *OR Spektrum* , 3 - 16, Vol. 11.

Hildum, D. (1994). *Flexibility in a knowledge-based system for solving dynamic resource-constrained scheduling problems.* Amherst: University of Massachusetts (Department of computer science).

Ho, W., Ho, G., Ji, P., & Lau, H. (2008). A hybrid genetic algorithm for the multi-depot vehicle routing problem. *Engineering applictions of artificial intelligence* , 548 - 557, Vol. 21.

Kara, I., & Bektas, T. (2006). Integer linear programming formulations of multiple salesman problems and its variations. *European journal of operational research* , 1449 - 1458, Vol. 174.

Kim, J. (2009). Proposed methodology for comparing schedule generation schemes in construction resource scheduling. *Winter Simulation Conference.*

Kolisch, R., & Hartmann, S. (1999). Heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis. In J. Weglarz, *Project scheduling; recent models, algorithms and applications* (pp. 147 - 178). New York: Springer US.

Laporte, G. (1992). The vehicle routing problem: an overview of exact and approximate algorithms. *European journal of operational research* , 345 - 358, Vol. 59.

Law, A. (2006). How to build valid and credible simultion models. *Winter Simulation Conference*, (pp. 58 - 66).

Manuj, I., & Mentzer, J. (2008). Global supply chain risk management. *Journal of business logistics* , 133 - 155, Vol. 29, No.1.

Miliken, F. (1987). Three types of perceived uncertainty about the environment: state, effect, and response uncertainty. *Academy of management review* , 133 - 143, Vol. 12, No.1.

Montazeri, M., & van Wassenhove, L. (1990). Analysis of scheduling rules for an FMS. *International journal of production research* , 785 - 802, Vol. 28, No. 4.

Mulvey, J., Vanderbei, R., & Zenios, S. (1995). Robust optimization of large-scale systems. *Operations research* , 264 - 281, Vol. 43, No. 2.

Oberlender, G. (2000). *Project management for engineering and construction (2nd edition).* Boston: McGraw-Hill Higher Eduction.

Olde Kalter, M., Bakker, P., & Jorritsma, P. (2010). *(in Dutch) Woon-werkverkeer als drijvende krach achter groei automobiliteit.* Roermond: Kennisinstituut voor mobiliteitsbeleid.

Panwalkar, S., & Iskander, W. (1977). A survey of scheduling rules. *Operations research* , 45 - 61, Vol. 25, No. 1.

Peng Si Ow, & Morton, T. (1988). Filtered beam search in scheduling. *International journal of production research* , 35 - 62, Vol. 26, No. 1.

Pinedo, M. (2005). *Planning and scheduling in manufacturing and services.* New York: Springer Science+Business Media, Inc.

Pinedo, M., & Simchi-Levi, D. (1996). Heuristic methods. In M. Avriel, & B. Golany, *Mathematical programming for industrial engineers* (pp. 575 - 615). New York: Marcel Dekker Inc.

Ramasesh, R. (1990). Dynamic job shop scheduling: a survey of simulation research. *International journal of management science* , 43 - 57, Vol. 18, No. 1.

Schatterman, D., Herroelen, W., Van de Vonder, S., & Boone, A. (2008). Methodology for integrated risk management and proactive scheduling of construction projects. *Journal of construction engineering and management* , 885 - 893, Vol. 134, No. 11.

Shuttleworth, M. (2009, March 21). *Face validity*. Retrieved April 04, 2015, from Explorable: https://explorable.com/face-validity

Silver, E. (2002). *An overview of heuristic solution methods.* Calgary: Haskayne school of business, University of Calgary.

Surekha, P., & Sumathi, S. (2011). Solution to multi-depot vehicle routing problem using genetic algorithms. *World applied programming* , 118 - 131, Vol. 1, No. 3.

Vasile, D., & Vladut - Severian, I. (2013). Using probability - impact matrix in analysis and risk assessment projects. *Journal of knowledge management, economics and information technology* , 76 - 96.

Vidal, T., Crainic, T., Gendreau, M., & Prins, C. (2013). Heuristics for multi-attribute vehicle routing problems: a survey and synthesis. *European journal of operational research* , 1 - 21, Vol. 231, No. 1.

Wall, M. (1996). *A genetic algorithm for resource-constrained scheduling.* Massachusetts: Massachusetts institute of technology.

Wullink, G., Van Houdenhoven, M., Hans, E., Van Oostrum, J., Van Der Lans, M., & Kazemier, G. (2007). Closing emergency operating rooms improves efficiency. *Journal of medical systems* , 543 - 546, Vol. 31.

Xiaoxia Lin, Janak, S., & Floudas, C. (2004). A new robust optimization appraoch for scheduling under uncertainty: I. Bounded uncertainty. *Computers and chemical engineering* , 1069 - 1085, Vol. 28.