"Resource-Constrained Scheduling in the Engineering-Planning Phase of Civil Engineering Projects"

by

Izak Johann Potgieter

Dissertation presented for the degree of Doctor of Philosophy in Civil Engineering in the Faculty of Engineering at Stellenbosch University

Supervisor: Dr. G.C. van Rooyen

December 2015

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: .....

Abstract

"Resource-Constrained Scheduling in the Engineering-Planning Phase of Civil Engineering Projects"

I.J. Potgieter

Department of Civil Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Dissertation: PhD (Civil)

June 2015

Project planning and project scheduling have been the subject of scientific research for many years. Yet it is still common to hear of projects that wildly exceeded their budget and time estimations. Research indicates that large discrepancies still exist between project management theory and project management practice. It seems that the theories and techniques developed by academia have struggled to find their way into project management practice. There is a desperate need for industry specific research that can help bridge the gap between project scheduling theory and project scheduling in practice.

In very few industries are project escalations as commonplace and severe as in the civil engineering sector. Civil engineering projects that exceed their deadline and/or budget estimations seem to be the norm rather than the exception. This dissertation therefore focussed on project scheduling in the civil engineering context. One specific phase of the engineering process received focus, namely the engineering-planning phase. The scheduling requirements of the engineering-planning phase were investigated, and the attributes of high quality baseline schedules defined. The dissertation took a critical look at whether the scheduling techniques used in practice, or the scheduling techniques proposed by academia can fulfil the rigorous demands of the engineering-phase.

It became evident that both of these spheres fall short in this regard. Scheduling techniques used in practice are not optimised, and they ignore some of the most important project constraints. Academic scheduling techniques are not geared for projects of practical size, and the resulting schedules often lack resource-constrained critical paths. Neither of the two spheres pay

much attention to the uncertainties that is inherent to the engineering-planning phase.

A scheduling framework is introduced that aims to address these shortcomings. Two specific aspects required original work. Meta-heuristic scheduling techniques had to be adapted for projects of practical size, and slack centric resource allocation algorithms had to be developed. The relationship between the input parameters of meta-heuristics and project complexity is investigated, and a new slack maximisation resource allocation algorithm is presented.

This dissertation therefore not only provides new insights into the scheduling requirements of the engineering-planning phase, but it also offers project managers with new tools and techniques to generate high quality baseline schedules.

Acknowledgements

First and foremost, I have to express my gratitude to my supervisor and mentor Dr GC van Rooyen. You have taught me everything there is to know about Engineering Informatics, and it is difficult to imagine how I could have completed this dissertation without you. From having the patience to deal with my constant barrage of questions and excessively long meetings, to motivating me whenever an existential crisis (or my laziness...) threatened to bring me to a standstill, you have always just been a pillar of support in my life. I am forever in debt to you, and hope to one day be able to organise my thoughts with the same elegance that you display on a daily basis.

To my parents Sakkie and Linda Potgieter. I cannot thank you enough for all of the time and energy (and money!) that you have invested in my education. You have instilled within me a thirst for knowledge that can never be quenched, and for this I will be forever grateful. You have given me your love and support even when my artistic temperament got the better of me, and this in itself deserves a medal! My upbringing has been the most important ingredient in all of my achievements, and I love you both very much.

To my supervisors from abroad: Prof Martin Middendorf and Prof Wolfgang Huhnt. Thank you for giving me the opportunity to do research at your institutions, and for being such kind hosts. Living in Europe has been one of the most enriching experiences of my life, and you both have played an integral part in making this experience such a positive one.

To my business partner Louis Theron. For having the patience to work with me on a daily basis, and for being understanding when my energy had to be diverted from our business to my dissertation on a regular basis.

To all of the various institutions and companies that helped fund my studies. These include AECOM, the Harry Crossley Foundation and the OSP research fund from Stellenbosch University.

And last but not least, I have to express my deep gratitude to all of the countless researchers and individuals who provided the platform on which I could build my work. These individuals range from famous mathematicians from centuries past, to unknown programmers contributing to open source projects. At the risk of sounding trite, I can truthfully say that the words of Isaac Newton have never rang more true in my life:

"If I have seen further it is by standing on the shoulders of giants."

Contents

List of Figures

List of Tables

Nomenclature

ACO	Ant Colony Optimisation
AS-TSP	Ant System Travelling Salesperson Problem
BaSE	Baseline Scheduling for Engineering
BIM	Building Information Modelling
CCM	Critical Chain Method
CPA	Critical Path Analysis
CPM	Critical Path Method
MS-USM	Microsoft's Unconstrained Scheduling Method
MS-RCSM	Microsoft's Resource Constrained Scheduling Method
NC	Network Complexity
PERT	Project Evaluation and Review Technique
PSGS	Parallel Schedule Generation Schema
PSPLIB	Project Scheduling Problem Library
RAM	Resource Allocation Matrix
RCPSP	Resource-Constrained Project Scheduling Problem
RF	Resource Factor
RS	Resource Strength
SBSM	Stable Baseline Scheduling Method
SSGS	Serial Schedule Generation Schema
TOC	Theory of Constraints
WBS	Work Breakdown Structure
WIP	Work in Progress

Chapter 1

Introduction

In this chapter, project management and project scheduling are introduced to the reader. The gap between project scheduling theory and project scheduling in practice is illuminated, and a path towards unification is discussed. Project scheduling in the context of civil engineering will receive focus, and the main objectives of this thesis will be presented. An outline for the remainder of the document is also given.

1.1 Project management and project scheduling

Management problems have been the subject of scientific literature for many years. *Project management* is one branch of management that has been receiving a growing amount of attention in recent years (see Kerzner, 1998, Meredith and Mantel, 2000). It is becoming increasingly popular for companies to adopt a project-based structure when organising their work-flow. This has sparked an interest in project management from both practitioners and researchers. Leus (2003) informally defines a *project* as a 'unique undertaking, consisting of a complex set of precedence-related activities that have to be executed using diverse and mostly limited company resources'. He identifies the involvement of project management in both the selection and initiation of projects, as well as the operation and control of projects.

Project scheduling forms an integral part of project management. Project scheduling is concerned with the sequencing of project activities and the allocation of scarce resources. Scheduling plays an important role at almost all levels of project management. Schedules are essential for estimation, planning, execution and control. Of obvious practical importance, project scheduling has been the subject of scientific literature since the late 1950's. An impressive amount of literature is available on the subject, and the reader is encouraged to read Demeulemeester and Herroelen (2002) for an extensive overview. The first formalised scheduling techniques that were developed

ignored resource-constraints and only regarded the technological precedence constraints of project networks. However, it soon became apparent that resource demand and availability have to be explicitly modelled if realistic schedules were to be obtained. This realisation sparked an interest in the field of resource-constrained schedule optimisation. Several different optimisation problems have been formulated, and various exact and sub-optimal procedures have been proposed as solutions to these problems. (Refer to Hartmann and Briskorn, 2008, for a comprehensive overview of the resource-constrained project scheduling problem and all of its variants.) Several project scheduling software packages have been developed over the years, and most commercial project management software provides some basic scheduling capabilities.

1.2 Theory and practice

Despite all of the attention that project management and scheduling have received, it is still common to hear of projects that far exceed their initial budget and deadline estimates. Several high-profile projects come to mind: The Brandenburg Airport in Berlin, Germany, 4 years behind schedule at the time of writing, The Channel Tunnel connecting the United Kingdom and France, exceeded estimated budget by 80%, and The Boston Big Dig, completed 9 years behind schedule and more than 190% over budget. These escalations are however not limited to large projects, with smaller projects frequently suffering a similar fate. Numerous publications have also reported on the matter, refer to Schonberger (1981), Group (1994), Winch (1996), and Yourdon (2003) for detailed examples.

Several authors have made an effort to classify the critical factors that determine the success or failure of a project. Noteworthy studies include the work of Pinto and Prescott (1990), Pinto and Mantel (1990), Belassi and Icmeli Tukel (1996), and Keil et al. (2003). Herroelen (2005) investigated the results of these studies, and came to the following conclusion: Workable project and contingency plans, as well as effective and efficient project monitoring and control procedures, are crucial to the success of a project. It is interesting to note that all of these factors are closely related to project scheduling, and that most of these aspects have received considerable attention from academia. This seems to indicate that the academic work done in the field of project scheduling has yet to find its way into daily project management practice. Herroelen (2005) confirmed this suspicion when he conducted an in depth study that documents the state of project scheduling theory and project scheduling in practice. He found that serious discrepancies still exist between the two spheres, with very little of the academic work being used in practice. It seems that project managers are reluctant to incorporate resource-constraints in the scheduling process, and that schedule optimisation techniques are rarely used. This becomes apparent when popular project management literature is

reviewed. Most project management text books dedicate a complete chapter to unconstrained scheduling techniques such as the Critical Path Method (CPM), but only a few sentences to resource-constrained project scheduling. The PMBOK® Guide (PMI, 2013), which is arguably the most popular project management reference used by project managers, dedicates a paragraph of only 13 lines (pg 179 - 180) to resource-constrained project scheduling. Herroelen (2005) notes that the PMBOK does not even recognise the essential difference between resource levelling (the smoothing of resource profiles over the project horizon) and resource-constrained optimisation (minimising the duration of a resource-constrained project). These publications tend to promote the idea that resource-constraints are somehow unimportant, and that unconstrained scheduling techniques such as the CPM are sufficient for planning purposes. This however stands in sharp contrast to the findings of academia. Some publications even go one step further and refute the usefulness of project scheduling entirely. The popular work of Goldratt (1997) is an example of this line of reasoning (pg. 217 – 221). Herroelen and Leus (2005a) critique these viewpoints, and illuminate the significant impact that resource-constraints can have on a project schedule. They stress the importance of selecting the appropriate scheduling techniques for the specific problem at hand. Their findings indicate that the accuracy of estimation and project planning can be greatly improved if these techniques were to be used more frequently in practice.

Reviews of commercial scheduling software tend to enforce the above mentioned ideas. Very few of the schedule optimisation problems addressed by academia are implemented by commercial scheduling software packages. Commercial scheduling packages tend to focus exclusively on the classic resource-constrained project scheduling problem and on resource levelling. Most of the other scheduling problems and optimisation techniques discussed in literature find no implementation in any of the commercial scheduling packages. The scheduling techniques used in commercial scheduling packages are mostly proprietary and their inner workings are rarely revealed to project managers. There is very little evidence of exact scheduling procedures being used in a commercial setting, with most popular software relying on basic priority rules for the generation of feasible schedules (Herroelen, 2005). Several studies have shown that these commercial scheduling packages are easily outperformed by the state of the art scheduling techniques discussed in literature (Kolisch, 1999, Debels and Vanhoucke, 2004). It is also rare to find any of the standard project scheduling terminology used in any of the commercial project scheduling software packages. Most commercial packages use their own jargon, and largely ignore the internationally accepted terminology used in scheduling literature (De Wit and Herroelen, 1990). Despite all of these factors, project scheduling software still remains popular amongst project managers. Surveys indicate that almost 80% of project managers use some form of project scheduling software, with Microsoft Project and Primavera Project Planner being the two most popular scheduling packages (Bounds, 1998). Studies have however indi-

cated that these scheduling tools are very rarely used for schedule optimisation or resource planning. Several surveys conducted in Europe indicate that most companies use scheduling software mostly for communication and presentation (De Reyck and van de Velde, 1999, Deckers, 2001). These studies also make it clear that project managers have limited knowledge of the software tools that they are using.

1.3 Bridging the gap

There is a desperate need for research that can help bridge the gap between project scheduling theory and project scheduling in practice. Several researchers have suggested key areas that still need to be addressed. This section will highlight the most important aspects of project scheduling that still require further research.

1.3.1 Applying problem specific scheduling techniques

A multitude of different scheduling problems have been identified, with academia proposing several techniques to solve these problems. It is important that project managers correctly identify the scheduling techniques that they require for the specific scheduling problem that they are attempting to solve. This is however not always an easy task considering the wealth of information that is available on the topic. Hierarchical planning frameworks have been proposed by several researchers to divide project planning into more manageable parts. Most researchers divide planning problems into three categories: strategic, tactical and operational. Some researchers also classify tactical/operational as another intermediate level. An example of a hierarchical planning framework for project organisations is shown in figure 1.1 (based on De Boer, 1998). Each level of the hierarchy has its own input data, planning horison, and output requirements. Different scheduling techniques need to be applied at each level of the hierarchy.

The scheduling techniques used at each of these levels will also be industry dependent. An additional positioning framework has been proposed by Leus (2003) to account for this (see figure 1.2). His positioning framework uses two key determinants to classify planning problems: The degree of variability in the work environment and the degree of dependency of the project (Leus, 2003, Herroelen and Leus, 2004). The variability is a measure of the uncertainty that results from a lack of information available at the time of scheduling. Uncertainty can stem from various sources, these include activity durations, availability of resources, scope of work and unclear objectives. The dependency is a measure of the extent to which the project is dependent on external influences. These influences can come from outside of the organisation, examples include dependency on subcontractors and suppliers. Dependencies

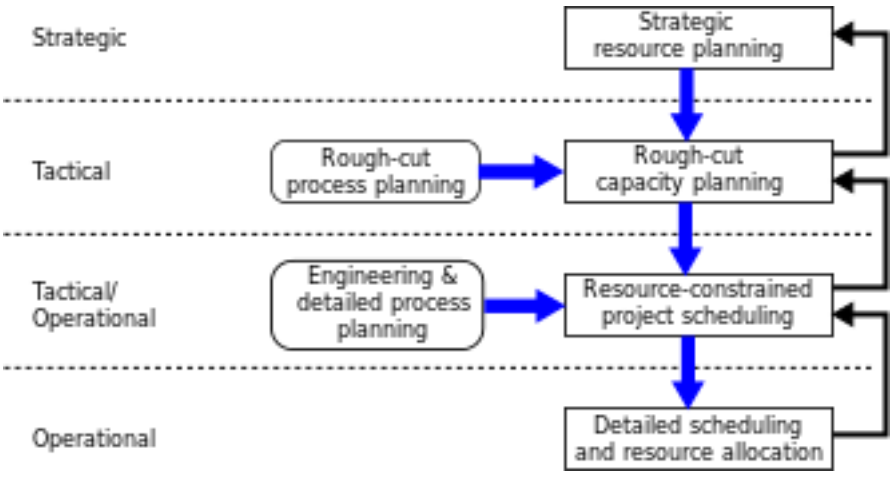


Figure 1.1: Hierarchical planning framework (based on De Boer, 1998)

can also come from within the company itself, for example shared resources amongst various projects. Applying this positioning framework at each level of the hierarchical planning framework, provides a universal mechanism for positioning the planning problems of almost all project based organisations.

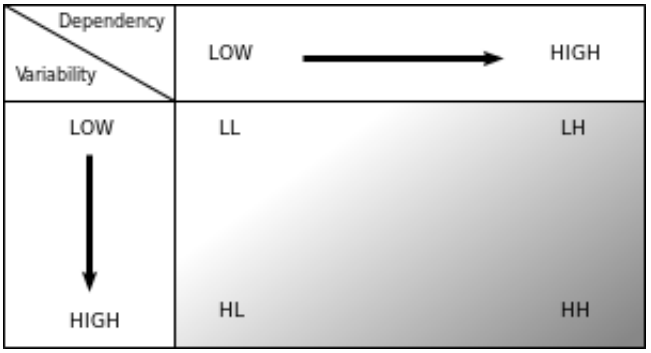


Figure 1.2: Positioning framework

Herroelen (2005) realised the importance of making project scheduling theory more accessible to project managers. As a first step towards this goal, he organised all of the different project planning problems identified by academia according to this hierarchical positioning framework. He highlights the scheduling techniques that are available to solve these problems, and identifies key areas that still require further academic attention. His work is an important step towards ensuring that project managers apply the correct techniques applicable to their specific problem. Industry specific research is however still required to help project managers classify and orientate their scheduling problems within this hierarchical positioning framework.

1.3.2 Scheduling in the face of uncertainty

It becomes clear from Herroelen's (2005) work that planning problems in the LL- and LH-environment of the positioning framework have received the bulk of academic attention. Projects that fall in these categories are typically executed in deterministic environments, and academia has proposed several scheduling techniques to address these problems. Scheduling problems that fall in the HL- and HH-environment have however received limited academic attention. These problems are subject to uncertainty, and very few scheduling techniques can adequately account for this. Several of the solutions that have been proposed do not incorporate resource-constraints at all. Ignoring these constraints will however lead to inaccurate schedules in almost all cases. The Project Evaluation and Review Technique (PERT) is an example of such a technique that is popular in practice. Unfortunately the few approaches that do attempt to incorporate resource-constraints also have some serious shortcomings. The Critical Chain Method is arguably the most popular of these methods, but researchers have pointed out that it seriously oversimplifies the problem (Herroelen and Leus, 2001, Herroelen et al., 2002). Some promising work has been done in the field of proactive/reactive scheduling techniques in recent years (see Leus, 2003, Leus and Herroelen, 2002, Van de Vonder et al., 2005). Instead of attempting to explicitly model uncertainty, these methods focus on generating stable schedules that are not sensitive to disruptions. This is achieved by intelligent resource allocation that optimises the slack distribution of a schedule. These methods have only started to appear in literature in recent years and have yet to reach maturity. Considering the inherent variability common to most real-life projects, it makes sense to dedicate more research effort to scheduling under uncertainty.

1.3.3 Identification of resource-constrained slack and critical paths

The reluctance of project managers to adopt resource constraints in the planning process has been noted by academia. Several factors have been identified which seem to contribute to this problem. Bowers (1995) argues that central to this problem lies the fact that it is difficult to interpret the results of a resource-constrained analysis. For the unconstrained case, the Critical Path Method (CPM) helps project planners to identify the slack of activities as well as critical paths. This is invaluable management information. Project managers can focus their attention on the critical path as well as activities with limited slack. Slack values can be carefully monitored during project execution, and project managers can assure that the best resources are assigned to critical activities. Unfortunately the familiar concepts of slack and critical paths are not readily available after resource-constrained schedule optimisation has been done. With no slack or critical paths to manage, project managers

are forced to micro-manage the project schedule, attempting to execute every activity exactly as planned. This might be feasible for small projects, but it quickly becomes an unrealistic approach as projects grow in size. As a result, project managers tend to avoid resource-constrained optimisation, relying on the CPM to provide them with critical paths and slack values. Unfortunately these values will almost always be inaccurate due to the CPM's inability to account for resource-constraints. Project managers are therefore basing their management decisions on incorrect data. For resource-constrained optimisation to really gain momentum, it must be possible to identify slack and critical paths in resource-constrained schedules. Some work has been done in this regard in the late 1980's and early 1990's (see Willis, 1985, Ragsdale, 1989, Bowers, 1995). Researchers showed that it is possible to identify critical paths and slack in resource-constrained schedules with minimal effort. Bowers (1995) showed that resource allocation plays a central role in this process. Unfortunately these ideas never reached academic maturity. Some much needed work still has to be done to bring these concepts to their full potential.

1.3.4 The need for scheduling software

Another major factor that is prohibiting the widespread use of sophisticated scheduling techniques is the lack of appropriate software tooling. Schedule optimisation cannot be done without the aid of a computer. Project managers typically do not have the time nor technical expertise to implement the schedule optimisation procedures that they require. Open source scheduling frameworks need to be developed that can be used by project managers. Several of the most important schedule optimisation algorithms need to be implemented before they will really gain popularity in practice.

1.4 Scheduling in civil engineering

As previously mentioned, it is very common to hear of projects that fail to meet their initial time and budget estimations. In very few industries is this problem as severe and commonplace as in the civil engineering industry. Apart from the high profile projects mentioned in section 1.2, most readers will be able to recall a civil engineering project close to home that failed to meet its planned objectives. Whether it be the construction of a shopping center, the design of a new bridge, or the maintenance of a road, it is typical for these projects to finish behind schedule and over budget. With such a vast amount of research stressing the important role of planning and scheduling in the success/failure of a project, it makes sense to study these factors in the context of civil engineering. Several questions still remain unanswered: What is the role of scheduling in each phase of a civil engineering project? What techniques are being used by project managers to generate these schedules?

Does academia propose different techniques for solving these problems? Is there a need for additional research in this field? Can commercial software cope with the scheduling needs of the civil engineering industry?

Answering all of these questions for each phase of a civil engineering project is beyond the scope of a single study. Instead, the different phases of a civil engineering project will have to be identified, and these phases will have to be addressed one at a time. Civil engineering projects can be broadly divided in two phases: The engineering phase, and the construction phase. The engineering phase refers to the design work involved in a project, and the construction phase refers to the physical implementation of the design. Typically the construction phase spans over a much longer time period than the design phase. However, this does not mean that the design phase cannot contribute to project delays. The construction phase is highly dependent on the information generated in the design phase, and it is important to deliver design documents in a timely manner. The construction phase can suffer serious delays if documents such as drawings or environmental impact assessments are not delivered on time. With fast-tracked projects becoming increasingly popular, this issue becomes even more important. In fast-tracked projects the construction phase begins before all of the design work is completed, and it is therefore imperative for the engineering phase to meet all of its estimated deadlines. Both the engineering phase and the construction phase can be further divided into two sub-phases: The planning phase, and the operational/execution phase. The planning phase refers to the preparation that needs to be done before work can commence, and the operational phase refers to the execution of the planned work. These phases are closely related, with the execution phase highly dependent on the information generated in the planning phase. Scheduling plays an important role in all four of these phases. Further investigation is required to ensure that the correct scheduling techniques are being employed in each of these phases.

1.5 Thesis objectives

The focus of this thesis will be on the engineering-planning phase of civil engineering projects. The engineering-planning phase is concerned with setting up a road map for the engineering-execution phase. During this phase, project managers need to determine what work needs to be done, what resources will be required, when work will have to be completed, and how much the project will cost. The deadlines and estimates set up in this phase will not only be used by the design team, but it will typically also be communicated to the client and to the contractor. In most cases, the company in charge of the design will be contractually obliged to meet these deadlines and estimates. Failure to do so could lead to penalties, delays and ultimately damaged reputations. It is therefore important for this project plan to be as accurate as possible.

This project plan is typically expressed in the form of a so called baseline schedule or pre-schedule. This schedule gives a project manager a clear indication of the scope of work to be completed, and the time-line involved. Milestone estimations, budget projections and resource planning are typically based on this schedule. The role of a baseline schedule is however not limited to projections and estimations, it also plays a vital role in project monitoring and control. Project managers will closely monitor the baseline schedule during project execution to ensure that the project stays on track. It therefore needs to provide project managers with performance measures and warning flags for when the project starts to fall behind. Critical paths and activity slack play an important role in this process.

Developing a high quality baseline schedule is the first step towards a successful project. It is therefore of utmost importance that project managers use sophisticated scheduling techniques that are tailored to their working environment. The scheduling needs of the engineering-planning phase has received very little academic attention, and as a result no guidance is available for civil engineers when they need to construct baseline schedules. Most engineers therefore blindly rely on commercial scheduling software to generate these baseline schedules, without a proper understanding of whether these tools fulfil their actual requirements.

The goal of this study is therefore to explore the scheduling needs of the engineering-planning phase, and to provide engineers with technical guidance for generating high quality baseline schedules. Four key objectives are identified:

Define the role of baseline schedules in the engineering-planning context: Baseline schedules are used intensively by project managers during the engineering-planning phase. It is important to clearly stipulate the purpose that these schedules are meant to serve during this phase. This can be achieved by studying the inner workings of both the design phase and the planning process of a civil engineering project.

Define the criteria for high quality baseline schedules: In order to evaluate schedules and scheduling techniques, it will be necessary to define the characteristics of a high quality baseline schedule. The worth of a scheduling framework can then be judged based on its ability to produce schedules that conform to these characteristics.

Evaluate the state of the art: With clearly defined criteria for high quality baseline schedules, it will be possible to provide a critical evaluation of the scheduling techniques used in practice, as well as the academic scheduling strategies available. This evaluation should deliver a verdict on the usefulness of these techniques in the engineering-planning phase.

Propose a scheduling framework: Project managers will need to be provided with a strategy to generate high quality baseline schedules. It will become clear from the literature review that neither academia, nor commercial scheduling tools can fulfil this need. A new scheduling framework, capable of producing high quality baseline schedules for the engineering planning phase, will therefore have to be developed. The core algorithms of this framework will have to be discussed, and appropriate tooling will have to be provided to make this framework practically viable.

The section that follows will define the outline of the dissertation, which should give the reader an idea of how these objectives will be achieved.

1.6 Dissertation outline

What follows is a brief discussion of each of the chapters included in this dissertation. This should provide the reader with an overview of the document structure, and it will hopefully help to orientate the reader as he/she progresses through this dissertation.

1.6.1 Chapter 2: Engineering planning and baseline schedules

In Chapter 2, the reader will be provided with an introduction to the engineering planning phase. Both the design phase and the planning phase will be discussed, highlighting the working environment and primary objectives of these phases. The role of baseline schedules in this context will also be discussed, and the criteria for a high quality baseline schedule will be defined for the engineering-planning phase of civil engineering projects.

1.6.2 Chapter 3: State of the art

Chapter 3 will serve as a literature review, discussing the state of the art in baseline scheduling methods. Both the scheduling methods used in practice as well as the scheduling methods proposed by academia will be discussed. An inquiry will be made into whether these methods are capable of producing high quality baseline schedules as defined in Chapter 2.

1.6.3 Chapter 4: Method overview

Based on the findings of the literature study, a scheduling framework will be proposed that caters for the unique needs of the engineering planning phase of the civil engineering industry. An overview of this framework will be presented in Chapter 4. The system consists of two core phases, namely the scheduling

phase and the resource allocation phase. Both of these phases along with their chief objectives will be introduced in this chapter.

1.6.4 Chapter 5: Resource-constrained scheduling

Chapter 5 will focus on the scheduling phase of the proposed framework. This phase is concerned with the makespan optimisation of resource constrained schedules. Even though a multitude of different algorithms have already been developed to solve this problem, only the most basic of these procedures have made their way into project management practice. This chapter offers insight as to why this might be the case. An attempt is made to bridge this gap, by providing practical enhancements to an existing Ant Colony Optimisation algorithm. Special attention is given to the selection of appropriate parameter values, and tooling is provided to better interpret the results of such an analysis. This chapter, along with chapter 6 provides most of the technical depth of this dissertation.

1.6.5 Chapter 6: A heuristic approach for maximising slack in resource constrained schedules

Chapter 6 will take a formal look at the importance of resource allocation in the generation of high quality baseline schedules. These allocations not only make it possible to identify slack and critical paths in resource-constrained schedules, but they also influence the total slack and slack distribution of a schedule. This chapter will formalise the resource allocation process, and discuss strategies for maximising the slack of resource-constrained schedules. A generic resource allocation algorithm will be introduced, and eight different heuristic allocation strategies will be explored. Detailed experiments will also be performed to show the practical implications that intelligent resource allocations can have on the planning of a baseline schedule. A brief discussion of the implementation and tooling of this phase will also be included in this chapter.

1.6.6 Chapter 7: Maintaining the process model

Baseline schedules are not only useful for projections and estimations, but they also play an important role during project execution. Although it is beyond the scope of this study to address the scheduling needs of the engineering execution phase, certain scenarios still need to be discussed that might compromise the integrity of the baseline schedule during project execution. Chapter 7 will look at these scenarios and explain the necessary precautions that management will have to take to maintain the integrity of the baseline schedule and the process model.

1.6.7 Chapter 8: Conclusion and recommendations

The final chapter of this dissertation will conclude the study and recommend potential areas for future research. The main findings of the study will be summarised, and specific aspects that still require further attention will be discussed. Potential research opportunities will also be identified, and a recommended plan of action will be proposed for each of these research areas.

Chapter 2

Engineering-planning and baseline schedules

Before scheduling techniques can be developed for the engineering-planning phase, it will first be necessary to define the work environment and to discuss the role of baseline schedules in this setting. This section will provide an informal introduction to the design environment and the planning phase of civil engineering projects. This will be followed by a detailed discussion of the role that baseline schedules play in the engineering-planning phase. To conclude this chapter, the characteristics of high quality baseline schedules will be presented.

2.1 Design environment

In the previous chapter, the engineering phase of civil engineering projects was defined as the design work that needs to be done before construction can commence. To understand the unique needs of this phase, it will be necessary to take a closer look at the working environment of a design office. Two aspects need to be discussed: The nature of the work, and the resources involved.

2.1.1 Nature of work

The main concern of a design office is the preparation of design documents required by the construction phase. Whether it be the design of a road, dam, building or bridge, the design process generally follows the same outline. Based on the specification of the client, preliminary models are developed for testing. These models are analysed and refined until they meet the standards prescribed by the codes of all applicable civil engineering disciplines. Once feasible designs have been established and approved, the final design documents can be drafted. These could include technical drawings, bending schedules, soil reports and several other key documents required for construction. The design cycle of

a civil engineering project requires input from various disciplines within the civil engineering sector. It is therefore typical to subdivide such a project into functional units called *work-packages*. These work-packages can then be further divided into executable units called *activities*. Activities represent the actual work that needs to be done, and they should have well defined boundaries. Each activity will have specific resource requirements, and there will exist interdependencies between most project activities. Take for example the design of the foundation of a building: If the geo-technical soil report is not complete, then foundation design cannot commence. Such interdependencies between project activities will be referred to as *technical precedence relations* from here onwards.

Even though each engineering project is unique, there exists many similarities between projects. Take for example the design of two unrelated office buildings. Even though the geometry will differ from one building to the next, the work-packages of each project will look very similar with many overlapping activities. Experienced engineers would therefore typically be able to provide accurate estimates for the activity durations, resource requirements and technical precedence relations of newly acquired projects.

2.1.2 Resources

The resources involved with design projects can be broadly classified into two groups: personnel and tools (Eygelaar, 2008). Personnel refers to the human resources executing the work, while the tools mostly refer to the software used to complete designs and reports. The personnel of a design office will typically consist of a large number of skilled individuals. These could include draftsmen, structural engineers, hydraulic specialists, geotechnical engineers and many more. The skill level of personnel members can vary from personnel-in-training to experienced workers. Personnel will typically be structured in functional departments such as the water department or the structural department. At any given moment, most personnel members will be involved with multiple projects. These projects won't necessarily have the same project manager, and the project teams might even consist of personnel from several different offices. The interdependency of design work, and the presence of multiple deadlines often times lead to a highly pressurised work environment. Personnel will be expected to complete work in a timely manner to ensure that projects stay on track. Deadlines for activities from unrelated projects might coincide, and personnel will often have to work overtime to reach their targets. Assigning personnel members to activities will therefore require careful consideration. It is important to ensure that critical activities are not assigned to individuals that are overloaded with work. Ideally the pressure of a project will be evenly distributed among its team members.

Since personnel members are often highly specialised, it can be difficult to acquire additional expertise on short notice. Project managers therefore

need to treat personnel limitations as a project constraint. Software tools on the other hand are much cheaper and easier to acquire on short notice. It is therefore not necessary to treat these tools as project constraints. For this reason, personnel resources will be the focus of this dissertation. The usage of the word *resources* will therefore refer to personnel for the remainder of this document. The term *resource type* will refer to the specific skill of a resource, for example a structural engineer.

2.2 Planning phase

The planning phase of an engineering project refers to the preparation work that needs to be done after a tender application has been successful, and before the actual design work can commence. The main goal of this phase is to develop a preliminary project plan. This project plan should provide an overview of the expected evolution of the project, and it is often referred to as the baseline schedule. Constructing this baseline schedule will require a detailed decomposition of the anticipated design process. Capturing and documenting this information produces a so called *process model*. This section will look at the work involved in setting up such a process model. Two questions need to be asked: What information will be available, and what information will have to be generated? The following sections will aim to answer these questions.

2.2.1 Available information

Since the planning phase follows the tendering phase, certain key project parameters will already have been agreed upon by both the client and the design company. These will include the scope of work, the project deadline and professional fees. The client might allow for some minor flexibility regarding these parameters, but generally they would be fixed contractually.

Before the tendering phase starts, top level management would also determine how many office resources can be offered to the new project. They achieve this by evaluating the capacity levels of all of the different resource types in the office. A portion of the remaining capacity can then be allocated to the new project, and these resource levels can be regarded as constraints in the planning phase. Note that it is *resource capacity* that is allocated to a project, not necessarily individual resources. In other words, if a company has 5 draftsmen, and a capacity of 2 draftsmen is allocated to a project, then it does not necessarily mean that two specific individuals are assigned to the project. It should rather be interpreted that at any given stage there should be the equivalent of two draftsmen available to work on the project. Which individual draftsmen will actually do the work will only be decided during project execution.

2.2.2 Information to be generated

In order to set up a project plan, a process model will first have to be developed that documents the work that needs to be done. A process model evolves in three phases (Eygelaar, 2008): In the first phase, all project activities will have to be identified. This will require the preparation of a complete Work Breakdown Structure (WBS). A WBS divides a project into its different work packages, and it documents the activities that need to be completed in each work package. Activity durations and resource requirements are also determined in this phase. These estimates require careful consideration, and it is normally done by a group of specialists from the applicable fields. Once this is done, phase two can commence. Phase two is concerned with the identification of the interdependencies that exist between project activities. Project managers need to identify the so called "has to be executed before" relation that exists in the set of activities. This is not always a straightforward endeavour, and various strategies exist to achieve this. The reader is encouraged to read the work of Eygelaar (2008) for a more detailed explanation of this phase. At the end of this stage a basic process model will be available. The final phase involves the review and evaluation of the process model. If the results are not satisfactory, then the process model will have to be adapted and re-evaluated.

Once it has been approved, the process model will be used to derive the baseline schedule of the project. In its simplest form, the baseline schedule simply provides project managers with the expected start and end dates of activities. However, project managers use a baseline schedule for much more than just this. The following section will give a detailed overview of the most important functions of a baseline schedule.

2.3 Role of baseline schedules

As stated in the previous section, the baseline schedule represents the preliminary project plan. Project managers typically don't expect to execute a baseline schedule exactly as planned, but they rather aim to defend a baseline schedule as best as possible. It should therefore be seen as a management tool rather than a project plan. What follows is a list of the most important functions of a baseline schedule:

1. Estimation of deadlines and milestones: During the engineering-planning phase the client will expect the project manager to set up important milestones and deadlines. These will not only allow the client to track the project progress, but it will also notify the construction company when specific design documents will be ready for collection. Once all parties have agreed on these deadlines, the design company will be contractually obliged to meet these delivery dates. Failure to do so will typically result in costly penalties, project delays and damaged reputations. It

is therefore essential for these deadline estimates to be as accurate as possible and that the project manager has a high level of confidence in meeting these dates.

2. Budget estimation and cash-flow projections: Both the client and the design company need to know what their expenditure will be for the first phase of the project. Both the total cost and the cash-flow requirements of the project need to be known. The baseline schedule provides a basis to calculate these costs and it ensures that the required capital can be secured in advance.

3. Resource planning: During the planning phase, the project manager will need to determine what the resource needs of a project will be. In order to secure appropriate resources in a timely manner, the skill requirements and resource distribution of a project will have to be clear. The resources types and the resource quantities of a project therefore need to be established before a project starts. The project manager must ensure that his most experienced resources are assigned to the most critical activities, and he needs to be assured of their availability. Since most of the personnel in a design office will typically be working on multiple projects, the project manager will have to book certain key personnel in advance to ensure their involvement in his project. The baseline schedule helps project managers establish resource demand over time, and it can be used for most resource planning needs.

4. Project monitoring and control during project execution: It is important for the project manager to track the progress of a project once the execution phase begins. Certain performance measures therefore need to be set up during the planning phase to achieve this. Project managers need to identify critical activities that require special attention, and they need warning signs to notify them when corrective action needs to be taken. Baseline schedules need to provide project managers with critical paths and slack values to assist with this process. This will allow project managers to focus their attention on critical activities whilst monitoring activity slack to identify potential problems.

5. Contingency planning and risk management: Uncertainty and risk is a reality in all engineering projects. The project manager needs to identify certain high risk areas in the schedule so that contingency plans can be put in place. These high risk areas can include resource bottlenecks and the convergence of several critical paths. The baseline schedule should not only assist the project manager with the identification of these high risk zones, but it should ideally attempt to minimise the existence of these situations. Having sufficient slack and time buffers in a schedule is one strategy to deal with this problem.

6. Visualisation and communication: All engineering projects have several key role players and stakeholders. These include the design company, the construction company, the client and investors. It is important that all of these role players have a common basis for communication and decision making. Key handover points need to be agreed upon, and cost and time projections need to be clear. The baseline schedule provides a road map for a project that can be used for this purpose. Presenting the schedule with a visualisation tool such as a Gantt chart will give all interested parties a clear picture of the expected development of a project.

2.4 Criteria for high quality baseline schedules

In this section the attributes and characteristics of high quality baseline schedules will be discussed. A baseline schedule can be considered of high quality if it can offer all of the above mentioned functionality to a project manager. Four key aspects are identified below which play a role in the generation of high quality baseline schedules.

2.4.1 Complete and accurate input data

Estimations and projections cannot be expected to be accurate if the input data of a schedule is inaccurate or incomplete. Even though certain detailed information will not be readily available at such an early stage of the project, the information that can be supplied should be as accurate and complete as possible. Accurate estimates for activity durations and resource requirements are required, and they can typically be provided by experienced professionals. It is also important that constraints such as activity interdependencies and limited resources are accounted for. Even though detailed information about specific resources will not be available in advance, capacity levels of resource types will typically be known. Failure to identify these resource constraints as input to the scheduling phase will result in incorrect estimations and inaccurate schedules. Care will have to be taken during the scheduling process to ensure that all of these resource constraints are respected. Baseline schedules that are of high quality will be based on accurate input data and they will not violate any scheduling constraints identified during the planning process.

2.4.2 Plan for uncertainty

All engineering projects will be subject to uncertainty and disruptions regardless of the accuracy of the input information. Several factors contribute to this problem: Activities might take longer than planned, resources might become unavailable, the project team might be under pressure to meet deadlines of

unrelated projects, the project scope might change etc. High quality baseline schedules will be able to absorb these disruptions so that the project plan remains largely unaffected. They achieve this by having certain safeguards in place to deal with uncertainties. Several strategies can be employed to meet this end. The intelligent distribution of slack and time buffers is one example of such a strategy that will safeguard a schedule against disruptions and uncertainties. Failing to include such safeguards in the baseline schedule will increase the risk of exceeding planned estimations and projections.

2.4.3 Critical paths and activity slack

High quality baseline schedules need to provide project managers with critical paths and activity slack values. Several of the most important functions of a baseline schedule discussed in section 2.3 will become meaningless if critical paths and slack values are unclear.

Identifying critical paths and activity slack is central to project monitoring and control. Having access to activity slack allows project managers to focus only on critical activities during project execution whilst monitoring the slack of near critical activities. Resources can be allowed to function almost autonomously, and project managers will only need to intervene if resources are starting to use up too much slack to complete activities. Without these slack values project managers will be forced to micromanage resources in an attempt to ensure that activities are executed exactly as planned. This might be possible for very small projects, but for projects of significant size this method of monitoring and control becomes will become near impossible.

Critical paths form a crucial part of resource planning. If activity slack is not known, then it becomes difficult for project managers to ensure that their best resources are working on the most critical tasks. This might result in inexperienced personnel working on tasks that might delay the project if they are not completed in a timely manner.

Critical paths are important for contingency planning and risk management. Corrective strategies need to be planned for the events which might disrupt activities on the critical path in any way. This might include booking extra resources in case of resource shortages or working overtime to make up for critical activities that took longer than planned.

It is important to note that slack and critical paths are characteristics of all project schedules. The difficulty however lies in correctly identifying these values in a schedule. In some rare cases, such as the schedules produced by the Critical Chain Method, it is trivial to identify slack. In most cases however, it is necessary to perform certain calculations post scheduling to calculate these values. An example of this is the Critical Path Method/Analyses (CPM/CPA) that can be applied to an unconstrained schedule to calculate slack and critical paths. Calculating these values for resource-constrained schedules is however not such a straight forward endeavour. Certain structural changes will have

to be made to the underlying process model during project scheduling to facilitate these calculations. More specifically, resource induced precedence edges will have to be introduced to the process model. These concepts will be further explained and developed as this dissertation progresses. At present it is however just important to keep in mind that the structure of a schedule should be in a state that accommodates the calculation and identification of critical paths and activity slack.

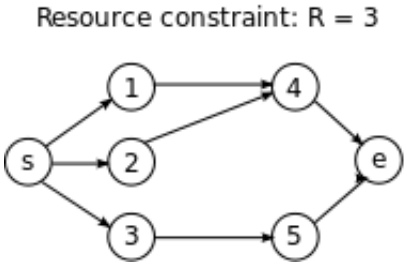
2.4.4 Optimisation

Several feasible schedules exist for any project. Determining the order in which activities are executed is not an easy task and it can greatly influence the outcome of a project. Project managers should always have specific objectives in mind when setting up the baseline schedule. Several properties of a schedule can be manipulated, and it makes sense to optimise those aspects most important to the project and environment under consideration. For the engineering industry, some of the most important factors include the project duration, the resource usage, and the amount of slack available in a schedule: Shortening the duration of a project will be financially beneficial to the client and it will ensure that the design company stays competitive; Efficient resource usage will allow the design company to take on more projects simultaneously; Increasing the amount of slack available in a schedule will take the pressure off the project team and lower the risk of missing deadlines. Choosing scheduling techniques that can optimise these factors would therefore greatly benefit both the client and the design company. Baseline schedules that have not undergone any form of optimisation cannot be regarded as high quality in an environment as competitive as the engineering industry.

Chapter 3

State of the Art

The previous section introduced the role of baseline schedules in the engineering-planning phase. The characteristics of high quality baseline schedules have been identified, and this naturally leads to the following question: Are project managers equipped with sufficient tools and techniques to help them generate baseline schedules that are of high quality? This section will set out to answer this question by investigating the state of the art scheduling tools used in practice, as well as the state of the art scheduling techniques proposed by academia. These tools and techniques will be evaluated according to the criteria of section 2.4. To facilitate this evaluation process, an example project has been created to showcase the results of the different scheduling methods. The basic process model of this example project is shown in figure 3.1. The figure shows the technical precedence network, activity durations and resource requirements of the project. Only one resource type is considered for the sake of simplicity. A resource constraint of 3 is imposed on this resource type. Note that activity s and e represent dummy start and dummy end nodes.



Activity	Duration	Resource Requirement
1	3	1
2	4	2
3	5	1
4	4	1
5	6	2

Figure 3.1: Example project

3.1 A Note on the Critical Path Method

Before discussing any scheduling techniques, it is necessary to stop for a moment to discuss the Critical Path Method (CPM). The CPM has become so

synonymous with project scheduling that it often becomes difficult to discern between the two topics. Even though popular project management literature promotes CPM as a scheduling method, one should not forget that its chief concern is to calculate critical paths and activity slack in a project network. It achieves this by calculating the earliest start dates and latest start dates of project activities by applying a simple path traversal algorithm to the project network (refer to section 6.2.2 for a more detailed discussion of the inner workings of CPM). The early start date of an activity represents the earliest possible moment in time that an activity can start if all of its predecessors are executed according to plan. The latest start date of an activity represents the latest point in time that an activity can start without delaying the project end date. Activity slack is calculated as the difference between the latest start and earliest start date of an activity, and the critical path of a project consists of all of the activities with zero slack. It is important to note that the CPM does not consider the scheduled starting times of activities when it calculates these slack values. Only the precedence constraints and the activity durations of the underlying project network are used for calculation purposes. If the CPM is therefore applied to a schedule whose underlying project network only contains technical precedence constraints, then it will produce unconstrained slack values, regardless of whether the schedule accommodates resource constraints. To successfully apply the CPM to resource-constrained schedules it will be necessary to alter the underlying process model during the scheduling process so that it includes not only technical precedence constraints, but also resource induced precedence constraints. This concept will be discussed in detail in chapter 6. For now it is however sufficient to note that the CPM will produce erroneous results if it is applied to a resource constrained schedule whose underlying process model only contains technical precedence constraints.

3.2 Scheduling in practice

Project managers are highly dependent on scheduling software for the generation of baseline schedules. For anything but small projects, manually scheduling projects is time consuming and error prone work. Most project managers therefore rely on the scheduling capabilities of commercial project management software tools for the generation of baseline schedules. These tools will dictate the input data that needs to be captured, the scheduling techniques which will be used, and the format in which schedules will be presented. In order to evaluate the scheduling techniques used in practice, it will be necessary to investigate the scheduling techniques used by these commercial scheduling software packages. Even though several commercial project management and scheduling packages are available, surveys and studies indicate that Microsoft Project (MS Project) and Primavera Project Planner are by far the most popular planning tools in the civil engineering sector (Liberatore et al., 2001,

Liberatore and Pollack-Johnson, 2003, Kastor and Sirakoulis, 2009). The construction industry seems to favour the use of Primavera, whereas MS Project dominates the engineering industry (J. van Huyssteen, Executive at AECOM, personnal communication, June 12, 2012). This section will therefore take a critical look at the scheduling capabilities of MS Project. Only the core scheduling techniques of MS Project will be analysed, without getting caught up in all of the different software features. Two aspects will be given special attention: MS Project's unconstrained schedules, and MS Project's resource levelling. These two complimentary techniques should actually be used in conjunction, but since resource levelling is largely ignored by more than 50% of project managers, these two topics will have to be treated separately (Liberatore et al., 2001, Kastor and Sirakoulis, 2009). To conclude this section, a verdict will be given on the ability of MS Project to generate high quality baseline schedules for the engineering-planning phase.

3.2.1 Unconstrained Scheduling Method

When setting up a new project, MS Project will automatically generate an initial project schedule for the user. This schedule corresponds to a classic early start schedule, and it is safe to assume that this schedule is generated with a technique similar to the CPM. This initial schedule can be derived with minimal input, requiring only the project start date, activity durations and technical precedence constraints from the user. This schedule does not account for any resource constraints, and is typically referred to as an unconstrained schedule. Applying this method to the example project will produce a schedule similar to the one shown in figure 3.2.

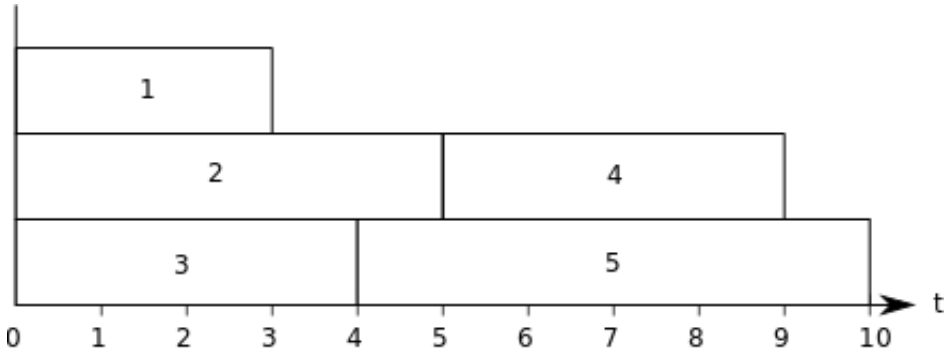


Figure 3.2: Unconstrained schedule

Since the source code of MS Project is proprietary, it is impossible to say with certainty whether these schedules are actually derived with the CPM. To avoid confusion, this scheduling technique will therefore be referred to as Microsoft's Unconstrained Scheduling Method (MS-USM) for the remainder of this document. Research indicates that a large percentage of project managers

tend to use these unconstrained schedules as baseline schedules (Kastor and Sirakoulis, 2009). Unfortunately these schedules do not meet the criteria of high quality baseline schedules. This becomes clear when MS-USM schedules are judged by the criteria defined in section 2.4:

3.2.1.1 Complete and accurate input data

The major drawback of the MS-USM is the fact that it does not account for resource constraints in the scheduling process. Since resource constraints are a reality in all engineering projects, these constraints have to be incorporated into the scheduling process. Resource constraints will impact the duration of a schedule as well as the starting times of activities. In most cases, the MS-USM will produce unrealistically short schedules with inaccurate start dates for activities. These schedules should therefore not be used for estimation purposes. Not only will this lead to erroneous deadline estimates, but it will also have a negative impact on aspects such as resource planning.

3.2.1.2 Plan for uncertainty

The MS-USM assumes that projects are executed in a strictly deterministic environment with perfect input data. As a result, the schedules generated by the MS-USM do not have any safeguards in place for possible project disruptions. Engineering projects are prone to disruptions, and there is always a degree of uncertainty involved in the planning of these projects. Failing to account for these uncertainties and disruptions increases the risk that a project will not go according to plan. The failure of MS-USM to adequately protect a schedule against such disruptions makes it a poor choice for the generation of stable baseline schedules.

3.2.1.3 Critical paths and activity slack

Performing a CPA on the MS-USM schedules will produce slack values and critical paths corresponding to an unconstrained project. This is however as expected, since MS-USM schedules do not incorporate any resource constraints. This lack of resource constraints will unfortunately produce overly optimistic results in almost all cases. The slack values produced by these schedules will typically be grossly inflated, and their critical paths can be incomplete or inaccurate. Resource constraints will inhibit activities from starting as early as the computed early start dates, and it will require that activities be completed before the latest finish dates. As a result, the critical paths of resource-constrained schedules will differ from those produced by unconstrained schedules. They might include additional activities, or even consist of a completely different set of critical activities. The inflated slack values produced by MS-USM schedules give project managers a false sense of security and it lets them focus on erroneous critical paths. Using the MS-USM to

generate baseline schedules will therefore have a detrimental effect on aspects such as project monitoring and control.

3.2.1.4 Optimisation

Since the MS-USM does not account for resource constraints, the resulting schedules will be in their most compressed form. Unless activity durations are altered, the duration of these schedules cannot be shortened and the slack values can not be increased. Optimising these two properties is therefore not applicable in this case. Compressed schedules will typically lead to high resource usage, and optimisation effort could have a positive impact on this property. The MS-USM does not attempt to optimise the resource usage of its schedules in any way.

3.2.2 Resource-Constrained Scheduling Method

It is obvious that MS-USM's inability to account for resource demand and availability severely detracts from its value as a scheduling tool. Limited resources are a reality in almost all projects and the effects of these restrictions need to be reflected in project schedules. To address this need, MS Project provides functionality to incorporate resources into the process model. Project managers can create resources of any type, and even set up resource calendars indicating the availability of these resources throughout the project. The resource demand of activities can be configured, and constraints on resource types can be set. This additional input might however lead to inconsistencies in the MS-USM schedules. Consider the schedule of figure 3.2. Activity 1, 2 and 3 are scheduled to be executed in parallel. This configuration will however create a resource demand of four resources. Since there are only three resources available for the duration of the project, these three activities can not be scheduled in parallel. The schedule is said to contain a *resource conflict*. In order to resolve this resource conflict, one of these activities will have to be shifted to the right to start at a later stage. Deciding which activity to reschedule is not a straightforward task, since each move will have a different downstream effect on the schedule. Resolving all resource conflicts of a project will produce a schedule that does not resemble the original MS-USM schedule. The schedule duration would typically be longer, and its resource usage would be different. Figure 3.3 shows how resource constraints influence the structure of the unconstrained schedule of figure 3.2.

MS Project provides two methods for resolving the resource conflicts in MS-USM schedules: The project manager can resolve them manually, or MS Project can do it automatically. Manually resolving resource conflicts involves shifting activities around by hand until no more conflicts are present in the schedule. This process is however tedious, and it will produce suboptimal schedules for all but small projects. The automatic procedure relies on a

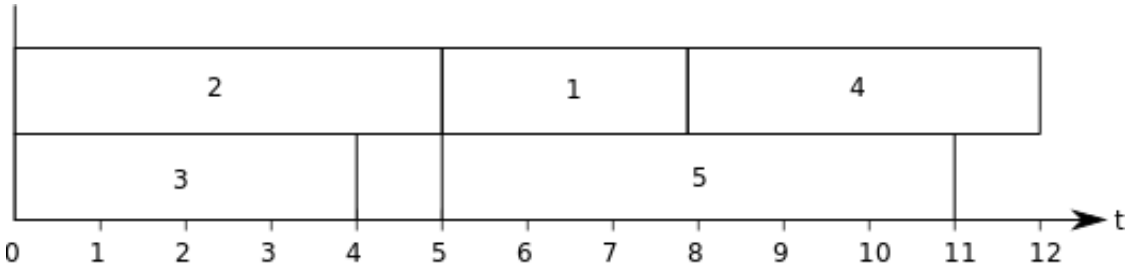


Figure 3.3: Resource-constrained schedule

scheduling algorithm to shift activities around until all resource conflicts are resolved. MS Project refers to this process as resource levelling. It is important to note that scientific literature differentiates between resource levelling and resource-constrained optimisation. Resource-constrained optimisation refers to the process of resolving resource conflicts whilst keeping the project duration to a minimum. Resource levelling refers to the process of resolving resource conflicts whilst keeping resource usage as even or smooth as possible. It is unclear whether MS Project's use of the term resource levelling is consistent with that of academia, or whether it also includes some resource-constrained optimisation. Unfortunately the official documentation does not shed much light on the matter either (Chatfield and Johnson, 2013, p.259). To avoid any further confusion between the two terms, MS Project's resource levelling algorithm will be referred to as Microsoft's Resource-Constrained Scheduling Method (MS-RCSM) for the remainder of this document. This section will discuss whether MS-RCSM is better suited for the generation of high quality baseline schedules than MS-USM. Once again, the criteria of section 2.4 will be used:

3.2.2.1 Complete and accurate input data

The MS-RCSM significantly improves the accuracy of MS-USM schedules by incorporating resource constraints in the scheduling process. These schedules will be much more realistic, and they will offer much more accurate estimations than MS-USM. If project managers ensure that all input data is accurate, then it can be concluded that MS-RCSM meets the criteria of having complete and accurate input data.

3.2.2.2 Plan for uncertainty

Unfortunately MS-RCSM still assumes deterministic input data, and the method does no more to protect schedules against disruptions and uncertainties than MS-USM does. It might offer improved estimations over MS-USM, but there is still the risk that these schedule estimates will become invalid due to disruptions and uncertainties. Since MS-RCSM does not factor in these concerns, it

can not be regarded as a good choice for the generation of high quality baseline schedules.

3.2.2.3 Activity slack and critical paths

There is one major disadvantage to using the MS-RCSM: MS Project cannot calculate the slack or critical paths in a schedule once MS-RCSM has been applied to it. If a CPA is applied to one of these schedules, it will simply produce the same results as for the unconstrained case. This is directly related to the fact that MS-RCSM does not alter the structure of the underlying process model during scheduling. It simply calculates appropriate start dates for activities, whilst displaying the results with a Gantt chart. The resource dependencies that the method sets out to resolve, are not reflected in the process model at the end of the scheduling process. The process model will therefore still only contain technical precedence constraints, and the CPM will produce slack values and critical paths applicable only to MS-USM schedules. Failing to correctly identify slack and critical paths in a schedule has a negative impact on several aspects of project management.

- Project control becomes nearly impossible if there are no critical paths or slack values to monitor.
- Managing risk and developing contingency plans rely heavily on the existence of critical paths.
- Activity slack is central to resource planning. Without these values it becomes difficult to ensure that the best resources are working on the most critical tasks

These factors severely detract from the usefulness of MS-RCSM for the generation of high quality baseline schedules.

3.2.2.4 Optimisation

Since the source code of MS Project is proprietary, it is difficult to determine which schedule optimisation algorithms are used by MS-RCSM. Several researchers have investigated the matter, and their findings seem to suggest that MS-RCSM employs basic priority rules to resolve resource conflicts. These priority rules deliver suboptimal solutions, and they are easily outperformed by the state of the art schedule optimisation techniques developed by academia (Kolisch, 1999, Debels and Vanhoucke, 2004). Herroelen (2005) notes that meta-heuristic algorithms such as ant colony optimisations or genetic algorithms could greatly improve the performance of commercial scheduling tools. MS Project will also have to provide project managers with more clarity regarding the objective function(s) that their scheduling algorithm is attempting to optimise. Baseline schedules have very specific properties that need to be

optimised, and it is important that the scheduling engine is tailored to these objectives.

3.2.3 Verdict

The scheduling capabilities of MS Project do not fulfil the unique needs of the engineering-planning phase. The two scheduling techniques offered by the tool are incapable of generating schedules that meet the demands of high quality baseline schedules. The inability of MS-USM to model resource demand and availability renders it incapable of providing project managers with accurate schedule estimations. It does provide project managers with valuable management information such as activity slack values and critical paths, but unfortunately these values are overly optimistic in almost all cases. To compensate for these shortcomings, MS Project allows for MS-RCSM to be applied to MS-USM schedules. Even though MS-RCSM is far from optimal, it does offer more realistic schedules than MS-USM. There is however one major drawback to using this functionality: critical paths and slack values are no longer available once schedules have been subject to MS-RCSM. The absence of activity slack and critical paths make these schedules very difficult to manage or control.

In an effort to overcome the individual shortcomings of each of these techniques, project managers might attempt to combine the two approaches: Using the more accurate resource-constrained schedules for estimation purposes, and the MS-USM schedules with slack and critical paths for management and control. This might seem like a meaningful approach, but since MS-USM slack values are inaccurate, it does not offer much of an improvement. The failure of both of these techniques to account for uncertainty further detracts from the usefulness of this approach[1].

Even though there are several alternatives to MS Project, none of these tools are equipped with the additional functionality required to produce high quality baseline schedules. Primavera does offer superior optimisation behaviour compared to MS Project (Liberatore et al., 2001), and also offers some PERT functionality in an attempt to deal with uncertainty. Similarly the scheduling package @Risk attempts to incorporate uncertainty in the scheduling process by running Monte Carlo simulations on schedules. Both PERT and Monte Carlo simulations attempt to answer certain "What if?" scenarios, but do not provide project managers with a single baseline schedule to defend. These methods give no indication as to which activities are critical and where project managers should focus their attention during project execution. The lack of resource-constrained critical paths in the schedules produced by these

[1] It should be noted that it is possible to use the Critical Chain Method (CCM) instead of CPM in MS Project via a third party plug-in, such as CCPM+. The CCM does attempt to compensate for uncertainties in the planning process, but it will become clear in the following section that this method also has several drawbacks.

method still severely detracts from their usefulness in the engineering-planning phase.

Its seems as if most commercial scheduling packages attempt to be too general purpose, with the hope of attracting a wide variety of industries. The end result is however generic scheduling tools incapable of addressing the specific needs of specialised fields such as civil engineering. There is a great need for industry specific scheduling software that can offer specialised scheduling functionality to project managers.

3.3 Academic scheduling

It should be clear that the scheduling techniques used in practice are not capable of producing high quality baseline schedules. This section will therefore investigate whether academia can provide any better alternatives. A multitude of different scheduling problems and techniques have been discussed in scheduling literature, and this section will focus on the techniques most applicable to the engineering-planning environment. In order to achieve this, it will be necessary to position the baseline scheduling problem of the engineering-planning phase within the hierarchical planning framework discussed in section 1.3.1. This will make it possible to identify the academic scheduling techniques most suitable to this scheduling problem. These techniques will be summarised and their ability to produce high quality baseline schedules conforming to the criteria of section 2.4 will be discussed. This section will conclude with a discussion of the additional research that still needs to be done to fulfil the scheduling needs of the engineering-planning phase.

3.3.1 Positioning the scheduling problem

Before the scheduling problem can be positioned, it will first be necessary to take a closer look at each of the levels of the hierarchical framework shown in figure 1.1:

At the top of the hierarchy is strategic resource planning. Not to be confused with the resource planning of a project, this level refers to the global resource planning of a company. Top level management needs to determine the global capacity levels of the organisation's resources and ensure that it is in line with the company's long term goals. If the company's resources will not be able to cope with the number of projects that the organisation would like to take on, then additional staff will have to be required, or the company will have to re-evaluate its subcontracting policies. If management finds that its resources will be under-utilised for the foreseeable future, then the organisation will either have to shift its goals to acquire more projects, or staff will have to be laid off. The planning horizon of this strategic resource planning level might cover anything from one to several years.

One level below strategic resource planning is the tactical level. At the tactical level, decisions are made regarding the acquisition of new projects. This level is closely related to the tendering phase of projects. Even though detailed project information will not be available at this level, management still needs to commit to several targets. Decisions will have to be made regarding the amount of company resources available for each individual project, and important project milestones and deadlines need to be set up. The planning horizon of the tactical level is typically between one and two years.

Following the tactical level, is the tactical/operational level. This level refers to the detailed planning of individual projects. Once a project has been accepted, more detailed information becomes available, and detailed project plans can be developed. Work packages can be broken down into individual activities, and the duration, interdependence, and resource demand of these activities can be determined. The resource levels determined in the tactical phase will be taken as constraints for the tactical/operational phase. Since the planning horizon of the tactical phase is quite long, it is still possible to acquire additional resource capacity at this level, and the resource levels of an individual project can be regarded as decision variables during this phase. However, since the planning horizon of the tactical/operational phase is considerably shorter, it becomes increasingly difficult and disruptive to acquire additional staff at such short notice. As a result, these resource levels need to be considered constraints for the tactical/operational phase. The project plans derived in this phase will therefore have to account for these constraints. The planning horizon of the tactical/operational phase can stretch from several months to a year.

At the bottom of the hierarchy is the operational phase. The operational phase refers to the execution phase of a project. Project managers need to determine what activities need to be completed on a week to week basis, and they need to ensure that the appropriate resources are working on these activities. The project progress must be closely monitored, and corrective action needs to be taken if necessary. The planning horizon of the operational phase covers anything from a week to a month.

It should be clear that the baseline scheduling problem of the engineering-planning phase best fits into the tactical/operational phase of the hierarchical framework. Now the scheduling problem needs to be placed within the position framework.

As stated in section 1.3.1, the vertical axis of the positioning framework represents the degree of variability in the work environment, and the horizontal axis represents the dependency of the project. Since almost all engineering projects are executed in a multi-project environment, the dependency of these projects is typically quite high. Most of the personnel in an engineering office will be involved with more than one project, and they will typically focus their attention on the project with the most immediate needs. This often results in activity delays due to resources being occupied with more pressing activities

from unrelated projects. To make matters worse, it will not even be clear how many projects a resource will be involved with once a project starts, since engineering companies acquire new projects on a continuous basis. It should therefore be clear that the high dependency of civil engineering projects are mostly a result of shared resources.

Uncertainty in civil engineering projects stems from various sources: The project scope may change, activities might take longer than planned, and resources can become unavailable at critical phases of the project. On first impression it therefore seems that civil engineering projects suffer from very high uncertainty. However on closer examination it becomes clear that several of these factors are not as severe as they seem. Consider the uncertainty related to the change in scope of a project. The client will typically be contractually obliged to compensate the design company for any extra costs incurred due to changes in project scope. It is therefore not necessary to incorporate the uncertainty pertaining to scope changes in the baseline schedule. The uncertainty of activity durations can be a result of several factors. Engineers might be unsure of the nature of work involved with an activity, leading them to underestimate the time required to complete the activity by a large margin. This scenario is however uncommon. Most design projects have many similarities, and activities are often similar in nature from one project to the next. Experienced engineers will typically be familiar with the type of work that needs to be done, and in most cases they will be able to give accurate estimations for activity durations. The uncertainty of activity durations are more often than not a direct result of the uncertainty regarding the resources that will be responsible for the execution of the activity. Inexperienced personnel might take much longer to complete activities than their more experienced peers, or the experienced workers might delay the duration of an activity due to being overloaded with work from unrelated projects. Since it is not clear who will be responsible for which activity at the start of a project, these variations in duration are typically not accounted for in the baseline schedule. This uncertainty is therefore actually a direct result of the high dependency of engineering projects. This is also true for the availability of resources. It is more likely that resources are unavailable due to commitments from other projects than for unpredictable reasons such as illness or resignation. Even though the variability in the work environment of civil engineering projects cannot be ignored, it leans more towards medium than high on the uncertainty scale.

It can therefore be concluded that the baseline scheduling problem of the engineering-planning phase falls in the high-dependency, medium-uncertainty region of the positioning framework. The work done by Herroelen (2005) suggests that projects with high-dependency should be scheduled using multi-project scheduling methods. When the uncertainty of these projects are high, stochastic scheduling methods and proactive strategies are proposed. These methods will be discussed in more detail in the following sections.

3.3.2 Multi-project scheduling methods

Multi-project scheduling methods attempt to incorporate the schedules of various projects into one large schedule that not only reflects interdependencies within each project, but also reflects the interdependencies that exist between projects. Since the interdependencies between engineering projects are typically resources, it would be possible to generate such a multi-project schedule by simply concatenating the process models of the individual projects and then feeding this information into a scheduling system. This would certainly produce a more accurate result than simply scheduling a single project in isolation, but unfortunately such a multi-project schedule cannot really be classified as a baseline schedule. It must be possible to manage individual projects in a modular fashion since neither the client nor the contractor would be interested in seeing information from unrelated projects in the baseline schedule. This would only serve as a distraction and it may even lead to confusion. This however does not mean that multi-project scheduling methods are worthless in the civil engineering context. They could prove to be valuable in the tactical level of the planning hierarchy. Top level management could make use of these methods to help them decide whether the company has the capacity to take on a new project. Unfortunately they are of little use for the generation of high quality baseline schedules.

3.3.3 Stochastic scheduling methods

Stochastic scheduling methods are concerned with minimising the expected duration of resource-constrained projects with uncertain activity durations. Uncertainty is explicitly modelled, typically by using distribution functions to represent the variation in activity duration. Selecting appropriate distribution functions is not always an easy task, and it often requires the experience of field experts. To minimise the expected duration of these projects, stochastic scheduling methods rely on so called scheduling policies to help project managers determine which activities should be started at various decision points throughout the course of the project. These decisions are based, in part, on the observed past, and can therefore only be made during project execution. Stochastic scheduling methods therefore do not construct a complete schedule at the start of a project, but the schedule is rather built up as the project progresses. Such scheduling strategies might be useful for short term planning or during project execution, but unfortunately the engineering-planning phase requires that a baseline schedule be set up before the project starts. These stochastic scheduling methods are therefore not applicable to the problem at hand, and they will receive no further attention.

3.3.4 Proactive scheduling methods

Explicitly modelling all of the uncertainties associated with a project is a near impossible task. Take for example the duration of an activity. There are numerous factors that can influence the duration of an activity: The amount of work that an activity requires might have been underestimated; The resources working on an activity might be too inexperienced to complete it in due time; The resources working on an activity might be too busy with other work to give the activity the attention it requires; There might be hold-ups due to incomplete information; Etc. etc. Attempting to factor in all of these possible disruptions in a schedule is not feasible. The best one can hope for is to set up a project schedule that is less sensitive to disruptions in the project input data. This is what most proactive scheduling methods set out to achieve. They aim to generate schedules that are able to absorb disruptions. They achieve this by distributing the project slack in an intelligent way, or by inserting time buffers at certain critical points in the project schedule. Disruptions can influence various properties of a schedule. These can include the starting times of activities, the project deadline, activity slack, and resource usage. Several objective functions can be formulated to ensure that disruptions affect these factors to a minimum.

One proactive scheduling technique that has gained popularity over the years is the Critical Chain Method (CCM), developed by Goldratt (1997). Several software implementations exist for the CCM, and it is actively used in practice. Even though the CCM has been heavily criticised by Herroelen and Leus (2005a), it will still be discussed here for the sake of completeness. Besides the CCM, very little work has been done in the field of proactive scheduling. One noteworthy study is the dissertation by Leus (2003) concerning the generation of stable baseline schedules. His scheduling technique and methodology will also receive attention in this section.

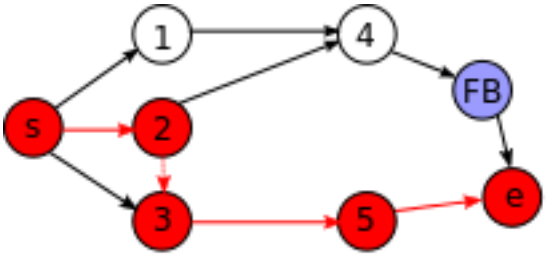
3.3.4.1 Critical Chain Method

The CCM is a scheduling method developed by Goldratt in the late 1990's, resulting from the application of Theory of Constraints (TOC) to project management. Since the publication of Goldratt's book Critical Chain in 1997 (Goldratt, 1997), the technique has gained in popularity, and it is arguably the most popular alternative to the CPM used in practice today. Numerous books (Newbold, 1998, Leach, 2000) and articles (Globerson, 2000, Patrick, 1999, Rand, 2000, Schuyler, 1997, 1998) have appeared in journal publications and popular project management literature, praising the method's innovative approach to project management. Several peer reviewed papers that have been published on the topic have however not been so generous in their appraisal of the method. Noteworthy studies include the work of Herroelen et al. (2002), Herroelen and Leus (2001) and Pinto (1999). Of these studies, Herroelen and

Leus (2001) provide the most in depth analyses of CCM. They scrutinized all assumptions, theories and results related to CCM, and back up up their findings with a full factorial experiment. The aim of this section is therefore not to provide a full analysis of the CCM, since there is very little to add to this conclusive study. This section will simply provide the reader with a brief overview of the fundamentals of CCM, highlighting its merits and pitfalls as documented by these studies. A critical look will also be taken at whether CCM can produce high quality schedules as defined in section 2.4, based on the findings of above mentioned studies.

Overview The CCM is a scheduling methodology built on the assumption that project disruptions and uncertainty are common to all projects. The CCM attempts to provide schedules with adequate protection against disruptions, whilst trying to avoid project delays caused by a phenomenon known as Parkinson's law (work expands to fill the time allowed (Gutierez and Kouvelis, 1998, Parkinson, 1957)). Several mechanisms are employed to achieve this end.

The CCM requires the same input as the classic resource-constrained scheduling problem, with one notable deviation: Activity durations used by the CCM will be significantly shorter than those used by more traditional scheduling methods. Goldratt (1997) argues that the duration of an activity can be accurately modelled by a probability distribution that is skewed to the right. To minimise the impact of Parkinson's law, Goldratt (1997) suggest that the median value of such a distribution should be used as the scheduled duration of an activity. This corresponds to activity durations that management have a 50% confidence level in achieving. Figure 3.4 shows the process model of the example project adapted for the CCM.



Activity	CCM Duration
1	2
2	3
3	3
4	2
5	4

Figure 3.4: CCM process model

CCM schedules are initially constructed by placing activities at their latest start times as provided by an unconstrained CPA. If the resulting schedule contains resource conflicts, then appropriate activities will have to be shifted back until all conflicts are resolved (Herroelen and Leus, 2001). The critical chain of such a schedule is defined as the longest unbroken chain of activities that determines the duration of the project. It is important to note that this critical chain takes into account both the technical precedence constraint

and resource dependencies that exist between activities. This critical chain is shown in red in figure 3.4.

To compensate for the shortened durations of the project activities, a *project buffer* is added to the end of the critical chain. Goldratt (1997) suggests that the buffer size should be 50% of the length of the critical chain. So called *feeding buffers* are also inserted wherever a non-critical chain of activities joins the critical chain. This is done in order to prevent disruptions of non-critical activities from propagating through to the critical chain. Resource buffers, usually in the form of advanced warnings, are also inserted between successive critical chain activities that are not executed by the same resource. These resource buffers are meant to act as a wake-up call to the resources that will be working on approaching critical activities. Figure 3.5 shows the resulting CCM schedule for the process model shown in figure 3.4. Note the use of buffers.

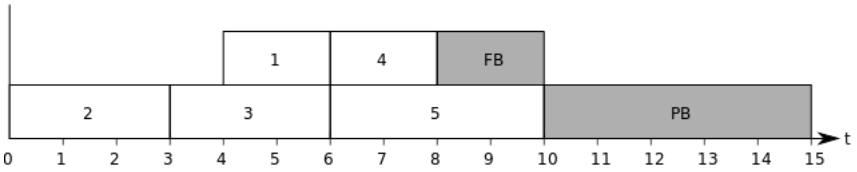


Figure 3.5: CCM schedule

Goldratt (1997) suggests that a so called *roadrunner* approach should be followed during project execution. In other words, activities should be started as soon as possible, regardless of scheduled starting times. This should be done for all activities, except those activities with no predecessors besides the dummy start node. The start dates of these so called *gating tasks* should coincide with their scheduled starting times. This is done in an attempt to decrease the system wide *Work In Progress* (WIP).

The CCM encourages project managers to monitor the consumption of buffer zones instead of the progress of activities. If there is still a sufficient portion of the buffer zone available, all is assumed to go well. Project managers only need to take corrective action once buffer consumption reaches a predefined critical point. This is an important shift of focus that fits in well with industries where micromanagement of activities is not practical.

Complete and accurate input data The CCM includes all of the most important constraints and parameters in the scheduling process. These include activity durations, technical precedence constraints, resource demand, and resource constraints. There is however concern regarding the accuracy of activity durations as specified by the CCM. Criticism is specifically directed at the fact that a fixed right skewed distribution function is typically used for all activities in a project. Herroelen and Leus (2001) state that this approach

"may prove to be inappropriate", whereas Dlowinski and Hapke (1999) reject it outright. Herroelen and Leus (2001) also point out that a fixed distribution function might not even hold for the execution period of a single activity. They also criticise the use of the median as the most realistic estimate of an activity's duration. They argue that mean values offer much safer estimates for activity durations, backing up their claims empirically. With so much uncertainty surrounding the accuracy of these activity durations, it becomes difficult to have a high level of confidence in the estimations and projections produced by a CCM schedule.

Plan for uncertainty The CCM's main advantage over traditional deterministic scheduling methods, is the fact that it makes a conscious effort to protect schedules against disruptions and project variability. Besides trying to capture the statistical variation of activity durations, the method also offers buffer zones as a means of protection against uncertainty and possible disruptions. This strategy bodes well with management, since they can focus their attention on these buffer zones, without having to micromanage each individual activity and resource. Only when activities start to eat away at buffers do management need to react by applying corrective strategies. At first glance this seems like a sound strategy, but unfortunately the CCM's implementation of these buffer zones is overly simplistic. Take for example the scenario where a disruption results in an activity penetrating a feeding buffer. It is very likely that this buffer penetration will result in resource conflicts that will have to be resolved. These conflicts could affect the critical path which the feeding buffer was meant to protect in the first place. Feeding buffers therefore offer a false sense of security to project managers. Herroelen and Leus (2001) provide an illustrated example of this scenario. Neither Goldratt (1997) nor any of the main proponents of the CCM offer any thoughts on the matter, and the issue remains unresolved. Besides these issues with feeding buffers, criticism has also been directed at the size of the project buffer. Herroelen and Leus (2001) have shown that the default 50% rule most commonly used for buffer sizing may lead to serious overestimations. They also reject Newbold's (1998) popular *root-square-error* method as an alternative to the 50% rule. It should therefore be clear that even though the inclusion of buffer zones is a step in the right direction, project managers should not be overly confident in CCM's ability to adequately protect schedules against uncertainty and disruptions.

Activity slack and critical paths The CCM correctly argues that the length of a project is not only a function of activity durations and technical precedence constraints, but also of resource demand and availability. The critical chain therefore not only incorporates the technical precedence constraints of a project, but it also reflects the resource dependencies that influence a project's duration. This important shift of focus was long overdue in the field

of project scheduling. Goldratt (1997) was however not the first person to identify the existence of critical chains. Wiest (1964) as well as Woodworth and Shanahan (1988) discussed the concept of so called *critical sequences* long before the existence of CCM. Unfortunately these studies did not gather much momentum nor industry attention, and the term *critical chain* has since become synonymous with CCM.

It is important to note that there can be more than one critical chain in a project schedule. When more than one critical chain is present, Goldratt (1997) suggests that the scheduler should simply pick any chain he/she desires. The selection of the critical chain will however have a significant impact on aspects such as feeding buffers, and will ultimately determine the final structure of the schedule. The critical chain of a schedule will also depend entirely on the scheduling method used to generate the initial schedule. A sophisticated optimisation algorithm will produce a very different chain of critical activities than for example a simple heuristic scheduling algorithm. Goldratt (1997) however sidesteps the issue, arguing that the choice of critical chain and scheduling method is unimportant. Several authors however disagree with this sentiment. Both Leach (2000) and Newbold (1998) state that the selection of the critical chain is an important strategic decision that could influence the outcome of a project. Herroelen et al. (2002) support this line of thinking, and they provide a detailed example to back up their claims. It becomes clear from these studies that the selection of the critical chain requires careful consideration. Literature on this topic is however scarce, and there is a need for additional research to assist project managers with this difficult topic.

Besides these issues with the identification of critical chains, there is also concern regarding the slack values of activities. The CCM schedules activities as late as possible, and they therefore don't have any slack besides the buffer zones that have been inserted in the schedule. Several inconsistencies regarding these buffer zones have however been revealed in the previous section, which make them an unreliable source of slack. Inconsistent buffer zones could lead to an overestimation of activity slack, which might result in a mismanagement of near-critical activities. There is a desperate need for a more reliable measure of slack in these resource-constrained schedules.

Optimisation The CCM promotes makespan minimisation as its number one objective, but does very little to actually achieve this goal. The CCM requires a resource-constrained schedule as starting point, but does not give any indication as to how this schedule should be set up. Resource-constrained scheduling is an NP-hard problem, and failing to optimise these schedules could have a drastic impact on the duration of a project. Most commercial implementations of CCM do not employ optimisation techniques when generating resource feasible schedules, but rather rely on basic priority rules to construct these schedules (Kolisch, 1999, Debels and Vanhoucke, 2004, Lib-

eratore et al., 2001). As explained in section 3.2.2.4, these schedules are far from optimal. Goldratt (1997) downplays the issue by stating that the impact of the scheduling method used, is small in relation to the uncertainty of the project. The factorial experiment conducted by Herroelen and Leus (2001) clearly show that this is not the case. Their study indicates that sophisticated branch and bound algorithms can have a significant impact on the makespan of CCM schedules.

The CCM also attempts to minimise the schedule's Work In Progress (WIP), by not letting gating tasks start before their scheduled starting times. Herroelen and Leus (2001) however show that the effect of this strategy is negligible, and that much better results can be achieved if sophisticated optimisation procedures are employed.

It can therefore be concluded, that the optimisation efforts of the CCM are not sufficient to produce high quality schedules.

Conclusion The few studies that have taken a critical look at the CCM all seem to be in agreement that the method has acted as an important eye opener to an industry where resource constraints and uncertainty are largely ignored. Most of the CCM's underlying assumptions are valid, but the danger lies in the overly simplistic way in which the method has been implemented. The method claims to incorporate resource constraints but in actual fact provides very little guidance or support for resolving resource conflicts. This becomes evident when one starts to examine the inner workings of mechanisms such as feeding buffers. Both the insertion and the penetration of feeding buffers can quite easily cause resource conflicts with which the CCM can not cope. When CCM schedules are evaluated against the criteria for high quality baseline schedules, it falls short in almost every aspect. Although it is without a doubt a step in the right direction, the CCM can not fulfil the unique demands of the engineering-planning phase of civil engineering projects.

3.3.4.2 Stable baseline schedules

Besides the CCM, very little academic attention has been paid to resource constrained scheduling subject to uncertain input parameters. One study that is however of interest, is the dissertation produced by Leus (2003). The aim of his study was to develop a scheduling framework capable of producing stable baseline schedules for industries prone to uncertainty. The method provides an innovative approach to scheduling under uncertainty, centred around the idea that resource allocations should be leveraged to optimise the slack distribution of a schedule. The reader is encouraged to read the research report of Leus and Herroelen (2002) for an in depth discussion of this method. A brief overview of the framework will be presented in this section, highlighting its advantages and disadvantages. Once again, the schedules produced by the framework will be evaluated against the criteria for high quality baseline schedules as defined

in chapter 2. For convenience sake, this scheduling framework will be referred to as the Stable Baseline Scheduling Method (SBSM) from here onwards.

Overview The chief objective of the SBSM is schedule stability. Leus (2003) defines stability as "a quality that is associated with a schedule when this schedule is able to suppress propagation of disruptions both within the individual project as well as towards other projects". He proposes that the stability of a schedule be measured in terms of the stability of the starting times of activities. If random fluctuations in activity durations cause little to no disturbance in the starting times of downstream activities, then a schedule can be said to be stable. The SBSM therefore aims to maximise the stability of a schedule as best as possible. A two phased strategy is employed to achieve this goal.

The first phase is known as the *scheduling phase*, and it is concerned with generating a resource feasible baseline schedule. This phase requires the same input as the classic resource-constrained scheduling problem. Activity durations are fixed, negating the need for cumbersome probability distributions. The SBSM prescribes no objective function or scheduling method for this phase, with the only restriction being that the resulting schedules must contain no resource conflicts.

The second phase is known as the *resource allocation phase*, and it takes the resource-constrained schedule generated in phase one as input. Whilst the scheduling phase is concerned with calculating *when* activities have to be executed, this phase is concerned with calculating *who* (or *which* resource) should be responsible for executing activities. The importance of resource allocations was first mentioned by Bowers in 1995. His work made it clear that resource allocations play a central role in both the calculation and distribution of resource-constrained slack. These concepts will be explored in detail in chapter 6 of this dissertation. For now it is sufficient to just be aware of the following key points: If resource assignments are done during/post scheduling, then resource dependencies can be reflected in the process model by simply inserting edges wherever individual resources are transferred between activities. This so called *resource flow network* can be combined with the technical precedence network to accurately represent all of the precedence constraints of a resource-constrained schedule. Applying a CPA to a schedule in this state will produce resource-constrained slack and critical paths. Figure 3.6 shows a feasible resource flow for the example project of figure 3.1. Note the resource-constrained critical path shown in red.

It is important to note that more than one feasible resource allocation/flow can exists for a single schedule. The selection of a feasible flow will influence both the total slack and the slack distribution of the schedule. Leus (2003) realised that he could exploit this fact to improve the stability of a schedule. The resource allocation phase of the SBSM therefore attempts to optimise the slack distribution of a schedule by means of intelligent resource allocation.

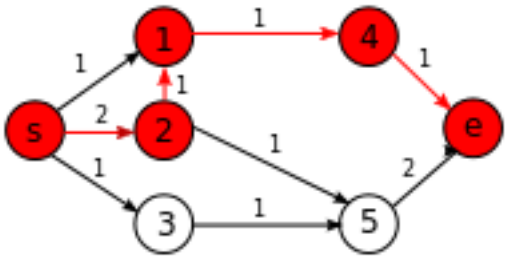


Figure 3.6: Feasible resource flow

This is achieved by employing a branch and bound procedure that explores different feasible resource flows in a schedule. For each feasible flow pattern, activity slack is computed, and tests are performed to measure how well the schedule can absorb the random disruptions of activity durations. When this phase is complete, activity slack and critical paths will be readily available, and the schedule's slack should be optimised for stability. Due to the complexity of this phase, SBSM was developed for projects that use only one resource type.

Complete and accurate input data The SBSM includes all of the most important constraints and parameters in the scheduling process. These include activity durations, technical precedence constraints, resource demand, and resource constraints. Unfortunately the SBSM can only cope with one resource type in its current state. Civil engineering project are typically multidisciplinary, requiring various different skills and resource types for a single project. Leus and Herroelen (2002) suggest that the resource type acting as the bottle neck in the system should be chosen if more than one resource type is present. They give no indication as to how this decision should be made. This decision is however not trivial, since it will have a major impact on the final structure of a schedule. The matter is further complicated in civil engineering projects, since there is typically no clear bottleneck resource type. Estimations and projections will also be inaccurate if only one resource type is considered. The SBSM will have to be adapted to cope with multiple resource types before it can be successfully applied to civil engineering projects.

Plan for uncertainty The SBSM deals with uncertainty in an implicit manner. Instead of attempting to explicitly model the variation of activity durations, the method aims to protect the starting dates of scheduled activities from possible disruptions, by means of intelligent slack distribution. Instead of artificially inserting slack into a schedule, the SBSM relies on a resource allocation algorithm to optimise the slack distribution of a schedule. This innovative approach avoids many of the issues involved with the artificial buffer

zones of CCM. Concerns regarding buffer sizing and buffer insertion are no longer relevant when the SBSM is used. Although several other authors have noted the importance of resource allocation in resource-constrained schedules (Bowers, 1995, 2000, Wiest, 1964, Woodworth and Shanahan, 1988), Leus and Herroelen (2002) were the first to exploit this strategy to combat uncertainty and disruptions.

The SBSM will be appealing to industries that require an execution schedule that closely resembles the original baseline schedule. Leus and Herroelen (2002) provide the example of airline scheduling, where it is of utmost importance that flights take off exactly when scheduled - no later and no earlier. Stable baseline schedules that protect the starting times of activities are a natural fit for these types of industries. The same can however not be said of the engineering-planning phase of civil engineering projects. If an activity is feasible for execution, then there is no point in delaying its execution so that it can correspond with its original starting date as specified by the baseline schedule. Doing this would only increase the risk of schedule overruns, since the activity might take longer than anticipated. If the necessary resources are available, then it makes sense to complete an activity as soon as possible. In engineering projects it is often the high workload of resources that leads to project delays and schedule disruptions. Ensuring that resources have sufficient slack could therefore go a long way in protecting a schedule against possible overruns. Instead of trying to keep the starting dates of activities stable, it might be more meaningful to employ slack maximisation strategies for these situations. No such resource allocation algorithms exist, and they will have to be developed if the SBSM is to gain any traction in the civil engineering sector.

Activity slack and critical paths One of the advantages of using the SBSM is the fact that resource constrained slack and critical paths can be calculated with minimal effort in schedules produced by the method. Once the resource allocation phase is complete, the resource flow edges can be fed back into the process model where they will represent resource induced precedence constraints. Applying a CPA to the process model in this state will produce resource-constrained slack values and resource-constrained critical paths. This is a major step forward, since it provides project managers with valuable management information which has been lacking in resource-constrained schedules for a very long time. The SBSM is firmly rooted in graph theory, and it is therefore easy to verify that these slack values and critical paths are indeed correct. This stands in sharp contrast to the CCM whose assumptions lack a strong mathematical foundation. The fact that most project managers are familiar with CPA further adds to the value of the SBSM.

Optimisation The SBSM allows for optimisation in both the scheduling phase and the resource allocation phase. For the scheduling phase, any re-

source constrained optimisation procedure can be used, in conjunction with any objective function. This is convenient, since a wide variety of sophisticated optimisation procedures have been developed over the years, and it makes sense to reuse some of these tried and tested procedures. Leus and Herroelen (2002) chose to treat the scheduling phase as a classic Resource-Constrained Project Scheduling Problem (RCPSP), where makespan optimisation is the number one objective. They made use of a branch and bound procedure developed by Demeulemeester and Herroelen (1992) to achieve this goal. This algorithm might however not be well suited for civil engineering projects, where the number of activities often times reach the hundreds or even thousands. Branch and bound procedures are typically computationally intensive, and they often cannot cope with problems of this size and complexity. It would make sense to use an algorithm more geared towards performance, for these situations. Meta-heuristic scheduling methods such as Ant Colony Optimisations, Genetic Algorithms, and Tabu Search Algorithms are all suitable candidates for this purpose.

During the resource allocation phase, the SBSM aims to optimise the stability of the schedule. It achieves this by employing a branch and bound procedure that explores different feasible resource flows in a schedule. Network flow calculations are unfortunately computationally expensive, and the performance of this algorithm will suffer as problem instances grow in size. The introduction of more than one resource type to the algorithm will further exacerbate the situation. Literature on the topic is scarce, and the author of this dissertation could not find any other resource allocation algorithms that alleviate this problem. As a first approach, it would be meaningful to develop heuristic resource allocation algorithms. These algorithms would be much more suited for problems of practical size, and their objective functions can be better adjusted to the needs of the civil engineering sector. These algorithms might not be optimal, but they would provide a good starting point for future researchers.

It can therefore be concluded that even though the schedules produced by SBSM are optimised, the methods used are not geared for problems of practical size and complexity. Fortunately the method's flexible structure would easily be able to accommodate alternative optimisation procedures and objective functions.

Conclusion Of all of the different scheduling methods discussed, the SBSM comes closest to matching all of the criteria of high quality baseline schedules as demanded by the engineering-planning phase. The method's use of resource allocation and resource flow to protect schedules against disruptions is nothing short of revolutionary. The resource-constrained slack and critical paths that the method produces as by-products, further confirms this view. It is unfortunate that the method does not account for more than one resource type, since

multiple resource types are commonplace in most industries. The stable activity start date objective function that the method employs is also not really applicable to the engineering-planning phase. Slack maximisation strategies could prove to be more meaningful in the engineering context. Regardless of these shortcomings, one very important point should still be taken from this method: Resource allocation plays a very important role during scheduling. Firstly, it allows for the identification of activity slack and critical paths in resource-constrained schedules. Secondly, it will influence the amount of slack and slack distribution in a schedule. Intelligent resource allocation algorithms could therefore be employed to protect schedules against disruptions and uncertainty. It makes sense to exploit this fact when attempting to generate high quality baseline schedules for the engineering-planning phase of civil engineering projects.

3.4 Chapter summary

This chapter took a critical look at the state of the art scheduling methods on offer for the engineering-planning phase of civil engineering projects. Both the scheduling techniques of commercial software, and the most applicable academic scheduling algorithms were evaluated. Unfortunately, none of these methods could satisfy the unique scheduling needs of the engineering-planning phase. The scheduling methods used in practice are incapable of producing high quality baseline schedules due to several key issues. The major concerns are the absence of resource constrained slack, inadequate protection against project uncertainty, and limited schedule optimisation. Academia offers superior solutions, but unfortunately none of these methods are 100% suited to the engineering-planning environment. The buffer centric approach of CCM may sound very appealing to management, but unfortunately the method suffers from some serious over simplifications. The SBSM offers a much more accurate alternative, based on sound mathematical principles. Unfortunately the method's objective functions and optimisation algorithms do not fit the high pressurised work environment of the engineering-planning phase where large project networks are common. It should therefore be clear that there is a need for a scheduling framework tailored to the needs of the engineering-planning phase. The remainder of this document will attempt to develop such a framework. This framework will share many similarities with the SBSM, but it will be customised to the unique needs of the engineering-planning phase. An overview of this framework is presented in the following chapter.

Chapter 4

Framework overview

4.1 Introduction

The previous chapter evaluated some of the most popular and most applicable scheduling techniques on offer to the civil engineering industry. Unfortunately none of these methods could satisfy all of the criteria required to generate high quality schedules for the engineering-planning phase. Even though these methods are not necessarily a perfect fit for the civil engineering industry, they do still provide valuable insights and concepts which have the potential to be further developed. This is particularly true for the SBSM. Of all the scheduling techniques discussed, this method comes closest to providing high quality baseline schedules for civil engineering projects. The method's innovative strategy for dealing with project uncertainty, and its ability to produce resource constrained slack, sets it apart from all of the other scheduling techniques discussed thus far. Besides these advantages, the method also offers several additional benefits that makes it a good choice for the civil engineering industry. The most important of these are:

Minimal user input: The SBSM requires the same input as traditional resource-constrained scheduling methods. No additional information is required for the method to adequately protect a schedule against possible disruptions. This should make the method very appealing for engineers, since this input is essentially the same as the input required by MS Project. It will therefore not be necessary for engineers to adapt their information capturing procedures in order to use this method successfully.

Modular structure: One of the great strengths of the SBSM is its modular structure. Even though the resource allocation phase depends on the schedule produced in the scheduling phase, the two procedures can essentially function independent from one another. This allows for great flexibility, since the objective function of one phase could be adjusted without affecting the other phase. Each of these phases could therefore

be geared towards the specific needs of the engineering industry. This modular structure also lends itself out to the reuse of previously developed algorithms. Several sophisticated scheduling algorithms have been developed over the years, and it makes sense to reuse these methods.

Deterministic input data: The SBSM implicitly deals with uncertainty by means of intelligent slack distribution. It is therefore not necessary to model the durations of activities with cumbersome probability distributions. These probability functions are not an exact science, and it would require engineers with a background in statistics to produce accurate values. The activity durations of the SBSM are simply modelled as fixed durations, corresponding to a realistic estimate of an activity's duration. This bodes well with engineering management, since it is in line with the estimation strategies currently used in practice.

Provides additional management information: Once both phases of the SBSM is complete, project managers will not only know when activities need to be executed, but they will also know who needs to be responsible for executing these activities. This is valuable management information that will be useful for both the planning and the execution phase of a project. None of the scheduling methods discussed in the previous chapter offer this benefit. As a result, project managers typically perform resource allocations without the aid of computer software. This process is both laborious and inefficient. The SBSM will relieve project managers of this burden.

Considering all of these factors, it makes sense to use the SBSM as a starting point when developing a scheduling framework for the engineering planning phase. The structure of the SBSM will be reused as a base on top of which this scheduling framework will be built. In other words, a two phased strategy will also be employed, consisting of resource constrained schedule generation, followed by a resource allocation phase with a slack based objective. The algorithms and objectives of these phases will however differ from those used by the SBSM, since they will have to be geared for the specific needs of the engineering planning phase. In the sections that follow, an overview will be given of the objectives and general approach of each of these phases. The scheduling framework will be referred to as the *BaSE* (Baseline Schedules for Engineering) framework for the remainder of this dissertation.

4.2 Phase 1: Resource-constrained scheduling

The first phase of BaSE is concerned with generating a schedule free from any resource conflicts. The input data required by this phase is in accord with the data that is available during the engineering-planning phase. This includes:

Activity durations: An accurate estimate will need to be provided for the duration of each project activity. Similar to the SBSM, these durations do not need to reflect the variable nature of these activities, but they simply have to represent a realistic estimate of the most probable duration of each activity. These durations can typically be provided by experienced engineers, or it can be derived from historical data.

Technical precedence constraints: The BaSE framework requires the technical precedence constraints that exists between activities. This so called "has to be executed before" relation in the set of activities can either be derived with the method described by Eygelaar (2008), or it can be specified explicitly. The latter approach is standard practice in engineering offices, while Eygelaar's (2008) method has mostly been limited to the academic sphere. Either of these methods can be used to provide the BaSE framework with technical precedence constraints, as long as the results are consistent.

Resource requirements: The resource requirements of each activity needs to be known before the scheduling phase can commence. It needs to be clear how many resources of each resource type every activity will require. Experienced engineers from applicable disciplines can typically provide project managers with these values.

Resource constraints: Resource limitations are a reality in all design offices, and these constraints need to be reflected in the schedule. The BaSE framework requires that the constraint of each resource type be fixed before scheduling can commence. It needs to be clear how many resources of each type will be available for the duration of the project. Since specific personnel members are normally not assigned to a project in the planning phase, these constraints will have to be based on the capacity level of each resource type assigned to a project. Top level management typically fix the resource capacity levels of individual projects during the tactical phase of the hierarchical planning framework discussed in the previous chapter (Leus, 2003).

With the input data in place, scheduling can commence. A multitude of different feasible schedules can be generated for a single project, and it is important to have an objective in mind when choosing a scheduling strategy. Several different objective functions have been formulated over the years for resource-constrained scheduling problems. The reader is encouraged to read the paper by Hartmann and Briskorn (2008) for an extensive overview of all of the variants of the resource-constrained project scheduling problem. They divide scheduling objectives into eight broad categories:

Robustness-based objectives: These objectives functions are concerned with improving the robustness or stability of a schedule. Since the resource

allocation phase of BaSE will address this issue, the scheduling phase does not need to consider this objective.

Objectives for rescheduling: It is not uncommon for projects to deviate from the initial project plan, and rescheduling is often required at some point during project execution. Rescheduling objectives are concerned with keeping the rescheduled project plan as close to the initial project schedule as possible. These objectives would be useful during the engineering-execution phase, but they are not applicable to the engineering-planning phase.

Objectives based on renewable resources: Increasing the size of the project team can often times significantly reduce the makespan of a project. This increase in resources will however raise the total cost of a project, and therefore requires careful consideration. Objectives based on renewable resources are concerned with assisting project managers with these decisions. Hiring additional personnel is typically not an option during the engineering-planning phase. Staffing levels need to align with the company's long term goals, and hiring policies are typically fixed during the strategic resource planning phase of the hierarchical planning framework discussed in the previous chapter. As a result, very little room is left for hiring additional personnel during the tactical-operational phase of project planning. The difficulty in acquiring skilled labour on short notice further exacerbates the problem. These objective functions are therefore not practical for the engineering-planning phase.

Objectives based on non-renewable resources: These objectives functions serve a similar purpose as those discussed in the previous paragraph, with the only difference being that they are focused on projects where non-renewable resources are dominant. Non-renewable resources refer to consumable resources such as building materials, whereas renewable resources refer to non-consumable resources such as personnel. Non-renewable resources are common during the construction phase of engineering projects, but they are typically not relevant to the engineering-planning phase.

Net present value objectives: Maintaining a positive cash flow is often times crucial to the success or failure of a project. The execution of activities and usage of resources will induce cash outflow, whereas capital investment and payments will produce cash inflow. The structure of a schedule plays an important role in the cash flow distribution of a project. Cash flow centric objective functions are referred to as net present value objective functions. In civil engineering projects, the costs associated with the construction phase are typically much higher than the cost of the design phase. Cash flow problems are therefore much

more relevant to the construction phase than to the design phase. Cash flow concerns should therefore not be the main driving force when setting up a baseline schedule for the engineering-planning phase.

Cost-based objectives: Besides the net present value objectives, several other cost based objectives have been formulated for the resource constrained scheduling problem. Scheduling problems that use cost-based objective functions typically allow the duration of activities to be shortened at an additional cost. The objective is then to reach the deadline of a project at the lowest possible cost. This process is also referred to as project crashing. These objective functions become relevant to the engineering-planning phase when the project team do not have the capacity to reach the deadline of a project. This situation will require personnel to work overtime at additional cost to complete the project. Cost-based objectives will have to be used to ensure that the cost of such a baseline schedule is as low as possible. It is however debatable whether such a crashed schedule should be used as a baseline schedule. A baseline schedule that contains crashed activities will place enormous pressure on the project team, and it should be a warning sign to project managers. These situations indicate that additional resources are required, or that certain parts of the project be subcontracted. These cost-based objectives might become more relevant during the engineering-execution phase if a project falls behind schedule, but it is best avoided during the engineering-planning phase.

Multiple-objectives: It often meaningful to have more than one objective in mind when setting up a project schedule. Several scheduling strategies aim to optimise a combination of several of the objective functions already discussed. The BaSE framework will follow a multi-objective strategy by optimising different objectives in the scheduling phase and the resource allocation phase. The respective phases will however be limited to one objective function each.

Time-based objectives: These objective functions are concerned with optimising some time-based aspect of a schedule. Examples of these objectives include optimising the deadline or milestones of a project, and minimising the lateness or tardiness of individual activities. In the engineering phase, the finishing dates of individual activities are not nearly as important as key milestones and the final deadline of the project. The client is typically quite lenient regarding the finishing dates of individual activities, as long as the key milestones and final completion date of a project is maintained. Failing to complete the engineering phase before the prescribed deadline will typically delay the construction phase and lead to costly penalties. Minimising the deadline or makespan of a project should therefore be the number one objective of the scheduling

phase. It will not only increase the probability of completing the project before its due date, but it will also allow the design company to take on more projects in a shorter amount of time. Employing such optimisation strategies will make a design company more competitive, whilst increasing profit margins.

It should therefore be clear that the main concern of the scheduling phase of BaSE will be to minimise the makespan of the project schedule. Makespan minimisation of resource-constrained projects is a well studied problem, and is referred to as the Resource-Constrained Project Scheduling Problem (RCPSP) in academic literature. Several exact and sub-optimal solutions have been developed for the RCPSP over the years (see Artigues et al., 2008, for an extensive overview). Some examples include branch and bound algorithms, genetic algorithms, tabu search algorithms and ant colony optimisations. Devoting more academic attention to this topic might therefore seem unnecessary. While this statement does have merit, it should be noted that virtually none of the more sophisticated scheduling algorithms have found their way into project management practice. Most studies that have investigated the matter are in agreement that the scheduling capabilities of commercial scheduling software can be greatly improved if more sophisticated scheduling algorithms were to be used (Kolisch, 1999, Debels and Vanhoucke, 2004, Herroelen, 2005). It is therefore necessary to understand what is hindering the acceptance of these algorithms in practice. Chapter 5 will investigate the matter, and also provide practical enhancements to an existing ant colony optimisation to make it more usable for the civil engineering industry. The selection of appropriate parameter values will receive special attention, along with development of specialized tooling.

4.3 Phase 2: Resource allocation

The scheduling phase of BaSE does not account for uncertainty or project disruptions in any way. Since makespan optimisation will often lead to a tightly compressed schedule, project disruptions could very easily disturb the completion time of a project. It is therefore of critical importance to build safeguards into these schedules in order to minimise the risk of project overruns. The ability of a schedule to absorb project disruptions is often referred to as *robustness* in academic literature. Schedule robustness is classified in one of two categories: quality robustness and solution robustness. Quality robustness of a schedule refers to the stability of the schedule makespan, whereas solution robustness refers to the stability of the starting times of activities (Kobylanski and Kuchta, 2007). Although some work has been done in this regard (see Van de Vonder et al., 2005, Al-Fawzan and Haouari, 2005, Kobylanski and Kuchta, 2007), the field has yet to reach academic maturity. A standard

measure of schedule robustness is lacking in most of the available research, and as a result, these studies tend to lack focus. Most of these researchers attempted to increase schedule robustness during the scheduling phase, where resource-constrained critical paths are absent. Measuring the impact of project disruptions on a schedule without critical paths is very difficult, since these disruptions will more often than not require a complete reschedule to resolve the conflicts that they cause. As a result, most researchers investigating the matter came up with their own measure of robustness to avoid this cumbersome reschedule. The SBSM developed by Leus (2003) is the only method that attempts to optimise schedule robustness post scheduling. Leus (2003) realised that different resource allocations/flows can have a significant impact on the total slack and slack distribution of a schedule. His method leveraged this fact in order to improve the solution robustness of a schedule. Having access to resource-constrained critical paths allowed Leus (2003) to accurately measure the impact of project disruptions, by comparing the early start schedules produced by applying a CPA to the process model before and after project disruptions. The result is a much more standardised and accurate approach to measuring the robustness of a schedule. As previously discussed, the stability of activity starting times is not nearly as important to the engineering-planning phase as the realisation of the project deadline. Resource allocation and slack distribution can however also be exploited to adequately protect the deadline of a project and improve the quality robustness. An optimisation process will have to be developed in order to find a resource allocation that best serves this purpose. Before this can however be done, a slack based objective will have to be identified that fits the engineering-planning environment. Unfortunately scientific literature is virtually void when it comes to the topic of resource allocation. Besides the stability objective that Leus (2003) investigated, no other slack based objectives have been identified for the resource allocation phase by academia. It is therefore necessary to look at the primary sources of uncertainty in the engineering-planning phase, and to derive a slack based objective based on this information. Uncertainty essentially stems from two sources: Uncertainty relating to project information, and uncertainty relating to project resources. Uncertainty relating to project information typically arises when the project scope is not clearly defined, or when the nature of work is unfamiliar. As discussed in the previous chapter, the design company should be able to claim for any project delays resulting from scope changes. The contract must therefore cover this aspect of uncertainty. Scenarios where the nature of work is unfamiliar, are not that common in civil engineering projects. The project activities of most engineering projects will typically be familiar to experienced engineers. Activity durations are therefore fairly accurate. Even though unique projects do exist, their unfamiliar activities would typically be limited to a handful per project. Uncertainty regarding project resources are normally a much bigger concern during the engineering-planning phase. The completion time of an activity is often times highly dependent on

the experience and workload of the resource responsible for executing it. Since it is not always clear who will be responsible for an activity that will only be executed in the distant future, uncertainty starts to creep into the activity durations. This is further complicated by the fact that individual resources are typically involved with multiple projects with different project managers. This issue, along with the fact that projects are acquired on a continuous basis, make it very difficult for project managers to judge the workload and potential performance of their resources. A mechanism that could reduce the pressure on the project resources could therefore go a long way in lowering the risk of missing project deadlines. This can be achieved by increasing the total slack of the project. This will give resources more time to complete activities, and it will reduce the number of critical activities. The objective of the resource allocation phase will therefore be to maximise the total slack of a project schedule. Maximising the total slack of a schedule has been attempted before by Al-Fawzan and Haouari (2005), but they did not approach the problem from a resource allocation point of view. With no critical paths to their disposal, they could not measure the impact of their objective function on the quality robustness of a schedule. Their work has been criticised by Kobylanski and Kuchta (2007), who attempted to standardise the way in which quality robustness is measured. Their schedules however still lack critical paths, and as a result they do not really improve on the situation of Al-Fawzan and Haouari (2005). This dissertation will attempt to maximise the total slack of a schedule by means of intelligent resource allocation. The resource induced precedence edges produced by the resource allocation phase will make it possible to clearly see the impact that this objective function has on the quality robustness of a schedule. Since no work has been done in this field, a heuristic algorithm will be developed as a first approach to solving the problem. Chapter 6 will set out to achieve this goal. Eight variants of the same heuristic allocation algorithm will be discussed, and experiments will be set up to measure the performance of these algorithms. The impact of these algorithms on the quality robustness of a schedule will also be tested on a large benchmark library for the first time.

The alert reader might however raise an objection at this stage: How can meaningful resource allocations be done during the engineering-planning phase if there is so much uncertainty regarding the future workload of personnel members at this stage of a project? Surely it would be more sensible to do resource assignments during project execution when more information is available regarding the whereabouts and workload of individual resources. While this is indeed true, it should be noted that the resource allocation phase does not necessarily imply that a specific individual be assigned to an activity. Proxy resources can be assigned to activities during the planning phase, and replaced with individual personnel members only when more information is available. Chapter 6 and 7 will further clarify this point. For now it is however sufficient to keep in mind that the resource allocation phase is not at odds with the information available in the engineering-planning phase.

Chapter 5

Resource-constrained project scheduling for the engineering-planning phase

5.1 Introduction

In the previous chapter it became apparent that the RCPSP adequately captures the requirements of the first phase of the BaSE framework. The RCPSP is concerned with the optimal scheduling of project activities, such that the project duration is minimised, without violating any of the resource or technical precedence constraints. (Refer to Hartmann and Briskorn, 2008, for an overview of the RCPSP and all of its derivatives.) Both the input parameters and the objective function of the RCPSP are therefore in line with the needs of the engineering-planning phase. A solution to the RCPSP will therefore also be a solution to the first phase of BaSE.

The RCPSP was one of the first scheduling problems to incorporate resource constraints, and it finds application in a wide variety of industries. Due to its practical appeal, several exact and suboptimal procedures have been proposed as solutions to the RCPSP. Unfortunately, the exact procedures often prove too complex for problems of practical size. At present, these exact procedures are useful for projects with up to about 60 activities, but after this computation times quickly become impractical. These algorithms would therefore be of little value to the civil engineering industry where projects can contain hundreds if not thousands of activities.

Several meta-heuristic procedures have however been formulated that are capable of producing high quality solutions for large problem instances. Examples include genetic algorithms, tabu-searches and ant colony optimisations. Although these procedures cannot guarantee optimal solutions, they do still provide sophisticated optimisation behaviour, capable of shortening a project's duration by a significant percentage. The computation times of these algo-

rithms are typically short, and they are not limited by the size or complexity of a problem instance. These algorithms therefore have the potential to be successful employed in the civil engineering industry.

Unfortunately these meta-heuristic algorithms have not found their way into daily project management practice. Most of the commercial scheduling packages in use offer solutions to the RCPSP, but these solutions are mostly derived using basic priority rules. Kolisch (1999) investigated the performance of these commercial scheduling packages, and found that on average they are suboptimal by 5%, with a range of 0% to more than 50%. It should however be noted that these test were performed on projects containing no more than 30 activities and four resource types. The performance of these basic priority rules can be expected to be significantly worse for larger projects. This proved to be the case when Debels and Vanhoucke (2004) compared these priority rules with state of the art meta-heuristic algorithms on projects with 300 activities. The meta-heuristics procedures outperformed the priority rules by a large margin, and proved to be capable of producing high quality solutions in a short amount of time. Large cost and time savings could be reported if these meta-heuristic algorithms find their way into civil engineering practice.

Meta-heuristic scheduling techniques have been around for more than ten years, with several researchers reporting on the potential benefits of these algorithms. It therefore seems odd that these techniques have not gathered more attention from project management practice. Several factors however still seem be hindering the acceptance of these algorithms in practice. Herroelen (2005) cites a lack of software support and limited know how of project managers as two factors contributing to this problem. Another complicating factor might be the fact that many meta-heuristic procedures are plagued by numerous input parameters. The selection of good parameter values is not always an easy task. Several meta-heuristic procedures have a random component, and it is often difficult to judge whether it is a change in parameter value or a random event that is influencing the solution quality. The process is further complicated by the fact that different problem instances might not respond the same to identical parameter values. Most researchers test their algorithms on the problem instances contained in *PSPLIB* (available at http://www.om-db.wi.tum.de/psplib/). This popular library contains a test bed of problem instances that consist of 30, 60, 90, and 120 activities with 4 renewable resource types. Most of the instances in the 90 and 120 problem sets have not been solved to optimality, and researchers often focus their attention on these problems. The result being that many of the algorithms and their parameters have been tailored to these specific problems and problem sets. Various industries need to schedule projects with more than 120 activities. In the engineering industry, projects can have as many as 1000 activities. This number might be even higher for construction projects. For these types of projects, the computation times of these meta-heuristic algorithms might become significant. Project managers do not necessarily have the time nor technical expertise to

experiment with various parameter settings. Academia has said very little about the selection of parameter values for such large projects. For these scheduling techniques to really gain traction in the civil engineering industry, project managers will have to be provided with guidelines for the selection of parameter values that can produce good solutions in a reasonable amount of time.

In order to successfully use one of these meta-heuristic procedures during the first phase of BaSE, it will be necessary to investigate the additional enhancements required to make these algorithms more usable for the civil engineering industry. Instead of therefore developing a new solution for the RCPSP, this chapter will study an existing meta-heuristic algorithm, with the goal of improving its usability. An Ant Colony Optimisation (ACO) that has proved to produce high quality solutions for the RCPSP will be the subject of discussion. A study will be made of the influence that parameter values can have on the solution quality of problems with different properties. The relationship between the pheromone evaporation rate and the convergence speed of a colony will receive special attention. Guidelines will be proposed for the selection of parameter values based on the properties of a specific problem instance. The practical requirements for a usable implementation of this algorithm will also be discussed.

The chapter is organised as follows: Section one will formally define the RCPSP. This will be followed by an overview of the ant colony algorithm, discussing both the approach and inner workings of the technique. Section three will take a look at the practical requirements that need to be fulfilled by the ACO to be useful in practice. Experiments and test are performed in section four, accompanied by a discussion of the significance of the results. Section five proposes an implementation and graphical interface for the ACO. Conclusions and recommendations are made in section six.

5.2 Definitions and problem statement

In this section formal definitions and notations will be introduced that will be used for the remainder of this dissertation. To start, a resource-constrained project will be defined, along with a feasible project schedule. This will be followed by a formal description of the objective function and problem statement of the first phase of the BaSE framework. An illustrated example is used throughout this section to clarify definitions for the reader.

5.2.1 Resource-constrained project and feasible schedule

A resource-constrained project consists of $J = \{0, 1, \ldots, n+1\}$ activities that have to be completed. Every activity $j \in J$ has a duration $d_j \in \mathbb{N}$. All activ-

ities have a non-zero duration, with the exception of the dummy activities 0 and $n+1$ for which $d_0 = d_{n+1} = 0$. A finite set of resources, $R = \{i, ii, iii \ldots\}$, is available for the duration of the project. Let $Q = \{R_A, R_B, \ldots, R_K\}$ be a partition of R according to resource types $A, B, C, \ldots, K$. The cardinality of each of these subsets represents the constraint of each resource type. For example, if $|R_I| = 4$, then four resources of type I are available for the duration of the project. Every activity $j \in J$ has certain resource requirements. These resource requirements can be conveniently represented by a $| J | \times | Q |$ matrix N of natural numbers. The row numbers represent the activity numbers and the column numbers correspond with the resource types. Let every element, n_{jI}, of the matrix represent the number of resources of type I required by activity j. The dummy start and end activities have zero resource requirements. Technical precedence relations exist between activities, and is represented by the set E. This set contains the pairs of activities between which there is a finish-start precedence relationship. It is assumed that graph $G(J, E)$ is acyclic. The dummy start activity $j = 0$ is the only activity with no predecessors, and dummy end activity $j = n + 1$ is the only activity with no successors. Figure 5.1 shows an example of a resource-constrained project.

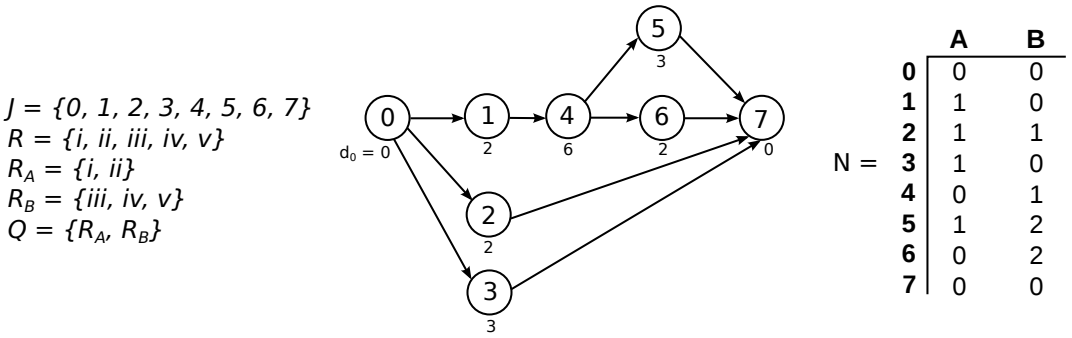


Figure 5.1: Resource-constrained project

A schedule S can be represented by a vector $\{s_0, s_1, \ldots s_{n+1}\}$, where s_j represents the starting time of activity $j \in J$. Let $f_j = s_j + d_j$ represent the finish time of activity $j \in J$. It should be clear that the start time and finish time of a schedule equals s_0 and f_{n+1} respectively. The makespan of a schedule is computed as the difference between its finish and start time. Let $J_t = \{j \in J \mid s_j \leq t < f_j\}$ denote the activities that are active during time instance t. A schedule is considered feasible if it satisfies the following constraints:

1. No activity $j \in J$ can be scheduled to start before all of its predecessors have been completed, i.e. $\forall (i, j) \in E : f_i \leq s_j$.

2. All resource constraints have to be respected, i.e. $s_0 \leq t < f_{n+1} \wedge \forall R_I \in Q : \sum_{j \in J_t} n_{jI} \leq |R_I|$

Figure 5.2 shows a feasible schedule for the example project.

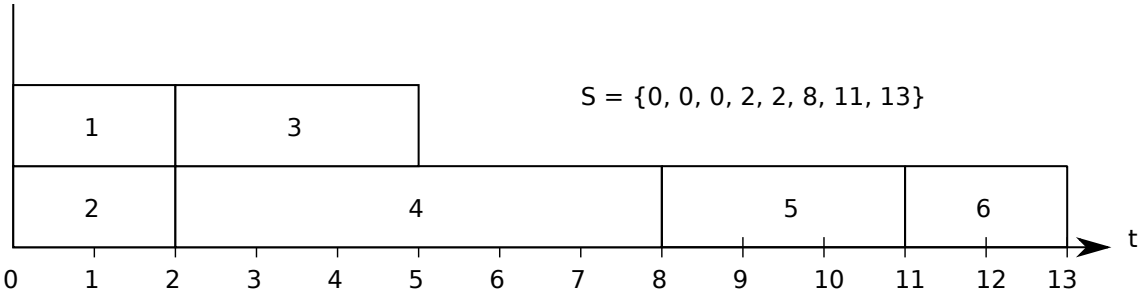


Figure 5.2: Feasible schedule

The problem statement of the first phase of BaSE framework (which corresponds to the RCPSP), can therefore be described as follows:

Problem statement 1: Given a resource-constrained project, find a feasible schedule S such that the project duration, $f_{n+1} - s_0$, is minimised.

5.3 Ant Colony Optimisation Overview

Many meta-heuristic algorithms have been proposed as solutions to the RCPSP. Several of these have proved to deliver excellent results. Some of the top performing algorithms include a self-adapting genetic algorithm Hartmann (2002), a simulated annealing algorithm Bouleimen and Lecocq (2003) and a tabu search algorithm Nonobe and Ibaraki (1999). For the purpose of this study, it was decided to focus on the Ant Colony Optimisation (ACO) that was developed by Merkle et al. (2002). This algorithm was chosen for several reasons:

1. The algorithm is one of the top performing meta-heuristics developed to date. It has been tested on the largest problem set of the *PSPLIB*, where it outperformed several of the most sophisticated meta-heuristics. Furthermore, the algorithm was limited to evaluate only 5000 schedules for each problem instance, showing its efficiency and potential for solving large problem instances.

2. A background in optimisation is not necessarily required to understand and use the algorithm. The algorithm is based on a process occurring in nature, and it is therefore not as abstract as several other optimisation algorithms. Several aspects of the algorithm map to real world phenomena, making it easy to visualise and understand. This is an important factor, since the project managers that will be using the algorithm will typically not have received any formal training in optimisation.

3. Several of the algorithm's input parameters have already been studied and fine tuned. The work that still needs to be done is therefore limited to only a few parameters.

In this section, a broad overview of the algorithm and its parameters will be given. Refer to Merkle et al. (2002) for a more detailed description.

5.3.1 Biological ant system

The ACO draws inspiration from the strategy that a colony of ants employs when searching for food. Being familiar with this process makes it easier to understand the optimisation algorithm.

A colony of ants always attempts to locate the food source closest to their nest. They achieve this by following a simple yet effective procedure. At first, all ants in the colony walk around at random searching for a food source. If an ant locates a food source, it will take a portion of the food back to the nest and mark the path that it follows with a trail of pheromone. Pheromone has two special properties:

1. Pheromone evaporates with time, so the further an ant has to walk, the more time the pheromone has to evaporate. The result is that shorter paths have a stronger scent than longer paths.

2. Ants are attracted by the scent of pheromone. If an ant finds a trail of pheromone, it will follow the trail to the food source and further enhance the scent of the trail by marking it with its own pheromone.

Given enough time, all of the ants in a colony should converge on the path that leads to the food source located closest to the nest.

This same strategy has been effectively applied to find paths in networks. Several artificial ants are allowed to traverse a graph, looking for paths with specific properties. Ants mark the edges of the graph with a trail of pheromone if a desired path is found. Similar to the biological ant system, pheromone evaporates with time, and ants are attracted to pheromone trails. Given enough time, all of the artificial ants should converge on the path with the most desired properties. The Ant Colony Optimization has been adapted to solve various different network based problems, but the basic idea always remains the same.

5.3.2 Schedule generation

Before the ACO scheduling algorithm can be discussed, it is necessary for the reader to understand how a feasible schedule is constructed. Two scheduling schemas have received the bulk of academic attention: The serial schedule generation schema (SSGS) and the parallel schedule generation schema (PSGS).

Merkle et al. (2002) investigated both of these procedures, and found that the ACO performed best in combination with the SSGS. This section will therefore only be discussing the SSGS. The reader is encouraged to refer to Kolisch and Hartmann (1999) should he/she be interested in understanding the inner workings of the PSGS. Note that the section that follows is largely based off the work of Merkle et al. (2002).

5.3.2.1 SSGS

The SSGS constructs a feasible schedule in n stages. It starts out with a partial schedule, containing only the dummy start activity scheduled at time 0. One activity is added to the partial schedule in each stage, completing the project schedule at the end of stage n. During each stage g, one activity is selected for scheduling from a set of eligible activities $j \in \xi(g)$. An activity is considered eligible for scheduling if it has not already been scheduled, and if all of its predecessors have been scheduled. The SSGS will always attempt to schedule activities as early as possible. In other words, an activity will be scheduled in the first resource feasible time slot available after the latest completion time of its predecessors. Algorithm 1 outlines the flow of SSGS.

Algorithm 1 Serial Schedule Generation Schema

```
1: procedure SSGS
2:     for g = 0 to n + 1 do
3:         compute eligible set ξ(g)
4:         select one j ∈ ξ(g)
5:         schedule j at earliest technical- and precedence- feasible start time.
6:     end for
7: end procedure
```

The SSGS can also be supplied with an ordered list of activities, in which case it will schedule the first eligible activity in the list during each stage. It has been proved that for every resource constrained project, at least one sequence of activities exists which will produce an optimal schedule if used by the SSGS (Kolisch and Hartmann, 1999).

5.3.3 Algorithm

The goal of Merkle et al. (2002) was to develop an ant colony optimisation that attempts to find a sequence of activities which will produce a good solution if used by the SSGS. This section will give a brief overview of the inner workings of this algorithm. Only the basic outline and flow of the algorithm will be discussed in this section. Section 5.5 will delve into more details of

the algorithm, by taking an in depth look at the specific parameter values and adaptations that can influence the behaviour of the algorithm.

5.3.3.1 Basic ACO

The ant algorithm developed by Merkle et al. (2002) was largely based on an ACO called the Ant System Travelling Salesperson Problem (AS-TSP) (Dorigo, 1992, Dorigo and Gambardella, 1997). The algorithm consists of a colony of ants that spawn g generations of m ants each. Each of the m ants in a generation is responsible for constructing one feasible activity list. An ant will build such an activity list in n phases, incrementally appending activities to the list. In each phase, an ant will have to decide which eligible activity should come next in the list. Ants will base their decisions on a combination of pheromone as well as heuristic information. Pheromone information is built up by ants that found good solutions in previous generations, and is conveniently represented in an $i \times j$ pheromone matrix τ. A high pheromone value τ_{ij}, indicates that previous generations of ants found good solutions when activity j was placed in position i in the activity list. Heuristic information is derived from a problem dependent heuristic that is typically fixed at the start of the algorithm. Heuristic information is denoted by η_{ij}. A normalised version of the latest start heuristic is used for the ACO developed by Merkle et al. (2002):

$$n_{ij} = (max_{k \in \xi} LS_k) - LS_j + 1 \tag{5.3.1}$$

where LS_j is the latest start time of activity j, calculated by applying the CPM to $G(J, E)$. [1]

The probability distribution over the set of eligible activities ξ is calculated as shown in equation 5.3.2:

$$p_{ij} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{h \in \xi}[\tau_{ih}]^{\alpha}[\eta_{ih}]^{\beta}} \tag{5.3.2}$$

Where parameter values α and β respectively control the relative influence that pheromone and heuristic values will have on the decision of an ant.

At the end of each generation, the pheromone matrix is updated according to the best solution found in the current generation, as well as the best solution found so far by the colony. This is known as an elitist strategy, since it will force the ants to search around previously found good solutions. Before this update can take place, a portion of the pheromone will first evaporate according to

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} \tag{5.3.3}$$

where ρ denotes the evaporation rate. This is be done to ensure that old pheromone information does not influence the behaviour of ants for too long.

[1] Please refer to section 6.2.2 to see how this calculation is performed.

For every activity in the project, a portion of pheromone is then added to element τ_{ij}, where i is the position of activity j in the activity list of the best solution found in the current generation of ants. This is also done for the activity list of the best global solution. Pheromone is updated according to

$$\tau_{ij} = \tau_{ij} + \rho \cdot \frac{1}{2T^*} \tag{5.3.4}$$

where $T^*$is the makespan of the solution under consideration. The algorithm terminates after a fixed amount of generations, or when the average solution quality has not improved for a predetermined number of generations.

5.4 Practical considerations

Optimisation procedures can rarely be used without any knowledge of the inner workings of the algorithm. This is also the case with the ACO. Due to the unique nature of every problem instance, the behaviour of the ACO might not always be 100% predictable. Project managers can therefore not blindly accept the final schedule that the ACO produces. The behaviour of the algorithm will have to be monitored, and the appropriate adjustments will have to be made if results are not as desired. The project manager will therefore require certain skills and have specific responsibilities during the schedule optimisation process. The most important of these are listed below:

- A basic understanding of the algorithm is required. Project managers need to understand the flow of the algorithm, as well as basic mechanisms such as pheromone evaporation and updating.
- Project managers need to be familiar with the most important input parameters of the algorithm. They need to understand the role of each parameter, and they need to be able to make educated predictions as to what might happen if certain parameter values are adjusted.
- Project managers need to be able to interpret the output of the algorithm. They need to be able to evaluate the convergence behaviour of the algorithm. If the algorithm converged prematurely, then parameter values will have to be readjusted so that a wider area of the solution space can be covered. Similarly, if the algorithm did not converge, parameters need to be adjusted so that the search can be intensified around good areas of the solution space.

Some guidance and tools will however have to be provided to facilitate project managers with the optimisation process. Some of the most important aspects are listed below:

- Project managers need to be provided with an implementation of the algorithm. Most project managers will not have any programming experience, and they cannot be expected to implement the algorithm on their own. Preferably the implementation should have a graphical user interface, where the project manager can enter the required input and evaluate the calculated output.

- Project managers will require some form of guidance when selecting parameter values for an optimisation. The ACO has several input parameters that need to be fixed. These values will influence the convergence behaviour and solution quality of the algorithm significantly. Since each project is different, with its own unique properties, it can be quite a difficult task to find the right parameter values for a specific problem instance. Experimenting with different parameter values is tedious and time consuming work, and project managers should be relieved of this burden as far as possible. Ideally good starting values should be generated for all input parameters based on the characteristics of the problem instance at hand. The project manager should then only have to perform minor adjustments to these values to achieve the desired output. Auto-generating good starting values might however prove to be difficult for the more complex input parameters. For these parameters, the project manager will have to be involved in the selection of starting values. This selection process will be significantly easier if the relationship between a parameter value, project complexity and solution quality is clear. At present these relationships are not clear for all input parameters, and further experiments are required to rectify this situation. The results of these experiments should facilitate the decision making process of project managers when selecting starting values for input parameters.

- The output and results of the algorithm should be presented in a format that allows the project manager to analyse the behaviour of the algorithm. The convergence behaviour of the colony should be clear and easy to track. This can be achieved in several ways, but preferably it will be done visually with the aid of a graph that tracks the solution quality over time.

- Ideally the process will be interactive. The project manager should be able to pause the algorithm, and analyse the behaviour of the colony at intermediate stages of a run. If results at intermediate stages are not satisfactory, then the algorithm can be reset with adjusted parameter values. This will allow the project manager to quickly understand the influence that parameter values can have on the behaviour of the colony, without having to wait for a run to finish.

In the sections that follow, the aspects mentioned above will be discussed in more detail.

5.5 Discussion of parameter values

Most meta-heuristic procedures are built on the idea that the optimal solution to a problem will be in the vicinity of other good solutions in the search space. The aim is therefore to target areas of the search space where good solutions are located. Parameter values play an important role in guiding the colony into good areas of the solution space. They not only influence the location where the colony should search, but they also influence the width and intensity of the search around these locations. Some parameters influence only the location or only the width of the search, whereas others have an influence on both aspects.

When Merkle et al. (2002) developed their ACO algorithm, they limited the number of schedules that could be analysed to 5000. This limitation meant their algorithm could only explore a small region of the solution space. It was therefore of utmost importance for them to make sure that their colony only explored very good areas of the solution space. As a result, they paid special attention to the parameters that influence the location and direction of the colony in the search space. Parameters influencing the width and intensity of the search received limited attention.

In a commercial setting, the number of schedules that can be evaluated will not be fixed, and it will be up to the project manager to decide how much time he is willing to dedicate to schedule optimisation. The goal is therefore to recommend parameter values to the project manager that can produce a good solution in a specified amount of time. It makes sense to focus the search around good areas of the solution space, regardless of the problem instance under consideration. For this reason, several of the recommended parameter values will be in accord with the suggestions of Merkle et al. (2002).

Deciding how wide an area should be explored around these locations will be largely dependent on the amount of time that is available for the optimisation and the complexity of the problem under consideration. The parameter values controlling this aspect will therefore require careful consideration. If the search is too wide, then the colony might not converge on a good solution in the allotted time. If the search is too concentrated around a certain area, then the colony will converge prematurely, possibly missing better solutions in the wider search area. Ideally the colony of ants should converge on the best solution only towards the end of the run. This gives an indication that the allotted time was put to good use by thoroughly exploring as large an area of the solution space as possible.

What follows is a discussion of the various parameter values influencing the ACO. The role of each parameter will be discussed, and guidelines will be given for the selection of good starting values for each of these parameters.

A summary of the work done by Merkle et al. (2002) will be given for the parameters that they have already investigated.

5.5.1 Balancing the influence of pheromone values and heuristic values

Parameters α and β control the relative influence that the pheromone values and heuristic values have on the decision of an ant. For most ACO algorithms these parameters are constant for the duration of the run. However, since a static heuristic is used, the danger exists that the ants could concentrate their search too much around the heuristic solution. This could lead to premature convergence and suboptimal solutions. Ideally the heuristic should guide the first few generations of ants to good solutions, but it should not hinder the ants from following good pheromone trails in later generations. With this idea in mind Merkle et al. (2002) decided to start with a relatively high heuristic value and to decrease its influence after each generation of ants. They set $\beta = 2$ at the start of the algorithm, and linearly decreased it so that it becomes zero after half of the generations have been completed. The pheromone influence was set to $\alpha = 1$ for the duration of the run. They found that this set-up produced superior results when compared to a constant value of $\beta = 1$ and $\alpha = 1$. It is therefore recommended that similar set-up with identical values be used. Only one slight modification is proposed: The beta value should influence a fixed number of generations, and not necessarily the first 50 percent of the generations. Their algorithm only analysed 850 generations, and the beta value therefore only affected the first 425 generations. If however 10 000 generations are analysed, the heuristic influence should not be present for the first 5000 generations. It is therefore recommended that the heuristic influence should not be present for more than 500 generations. This allows for more than enough ants to build up good pheromone paths, without allowing the heuristic to dictate the locations of the search for too long. This value should only be increased if the colony shows little to none convergence behaviour during this heuristic dominant period of the algorithm.

5.5.2 Number of ants per generation

The number of ants per generation controls the intensity of the search at a specific location. The more ants per generation, the more thoroughly a specific region of the search space will be explored. Since Merkle et al. (2002) analysed a restricted number of schedules, they only allowed 5 ants per generation. However, they did note that the solution quality can be improved if this number is increased. It is therefore recommended that at least 15 ants per generation be used. This number of ants will allow for an intensive search, without increasing the runtime of the algorithm by too large a factor. If time is not a factor, then this number could be increased to improve the solution

quality. It should however be noted that increasing the number of ants by a factor x will typically also increase the runtime of the algorithm by a factor x. Discretion should therefore be used when increasing the number of ants for large problem instances.

5.5.3 Discarding the elitist solution

As previously stated, two pheromone updates are done after every generation. One for the best solution in the current generation, and one for the best solution of the colony. The idea is that the ants should pay special attention to the area where the best solution was found. However, if the best solution of the colony does not change for several generations, then the strong pheromone trails around this solution might prohibit ants from exploring a wider search area. For this reason, Merkle et al. (2002) introduced the parameter g_{max}. This parameter limits the number of generations that an elitist solution is allowed to influence the pheromone updates. If the elitist solution does not change after g_{max} generations, then it will be replaced with the best solution in the current generation. They experimented with various different values ranging from 0 to 200, and they found that $g_{max} = 10$ produced the best results. High values of g_{max} led to premature convergence, and low values did not give the ants enough time to thoroughly explore the good areas of the search space. It is therefore recommended that $g_{max} = 10$ be used as a starting value for all problem instances. For complex problems where the solution space is really large, slightly higher values of g_{max} could help speed up the convergence behaviour of the colony.

5.5.4 Initialising the pheromone matrix

Before optimisation can begin, all entries in the pheromone matrix need to be initialised. Merkle et al. (2002) give no indication as to what values were used for the initialisation of their pheromone matrix. The initial values can influence the convergence behaviour of the colony, and should not be chosen at random. If the initial values are too high, then the effects of pheromone updates might not be significant enough, and the colony could take a long time to converge. If the initial values are too low, then the pheromone updates might cause the relative difference between entries to be too high, and the colony might converge early in the run. It is suspected that Merkle et al. (2002) initialised the pheromone matrix with

$$\rho \cdot \frac{1}{2T^*} \tag{5.5.1}$$

where T^* was taken as the duration of a schedule derived with the latest start heuristic. This is a strategy often used in other ant systems, and it should provide reasonable results in most cases. There is however a risk that the

heuristic solution might not be very good, which would result in the pheromone matrix being initialised with very small values. This could very easily lead to premature convergence and poor solution quality. This problem will be amplified for complex problem instances where heuristic schedules will be far from optimal. For this reason it was decided to modify the pheromone update procedure in a way that would eliminate the effect that the initial values have on the behaviour of ants . Instead of updating pheromone values according to

$$\tau_{ij} = \tau_{ij} + \rho \cdot \frac{1}{2T^*} \tag{5.5.2}$$

it is proposed that

$$\tau_{ij} = \tau_{ij} + a \cdot \rho \tag{5.5.3}$$

be used instead, with $a \in \mathbb{R}$. This configuration simply adds a constant amount of pheromone to the solution trail, without considering the makespan of the schedule under consideration. This set-up might seem odd at first, since it does not favour shorter schedules, but it does still lead to a convergent colony. This convergence will stem from the high pheromone values associated with the most popular solutions. The heuristic will initially lead ants to good solutions, increasing the pheromone of these areas in the solution space. The pheromone of these areas will then be further enhanced by the elitist solution, eventually driving the colony towards convergence. This set-up has two advantages: Firstly, if the pheromone matrix is initialised with a, then all elements in the matrix will always have a as a common factor, regardless of the number of times pheromone evaporates or is updated. Factorising equation 5.3.2 would therefore result in both the dividend and the divisor having a common factor of a, thus eliminating the impact that the initial pheromone will have on any probability calculations. Secondly, an element of the pheromone matrix will never exceed its initial value of a if exactly one pheromone update is applied after each evaporation phase. If a specific element of the pheromone matrix receives an update from every generation of ants, then its pheromone value will be equal to a at the end of the run. This is illustrated below, where an element initialised with a undergoes evaporation and a pheromone update in succession:

$$a \cdot (1 - \rho) + a \cdot \rho = a - a \cdot \rho + a \cdot \rho = a \tag{5.5.4}$$

The result is a very controlled environment, where pheromone values have an upper limit of a[2]. This stands in contrast to traditional ant systems where

[2]The ACO described developed in this dissertation executes two pheromone updates after each evaporation phase: One for the best solution in the current generation, and one for the best global solution. It is therefore not entirely accurate to say that the maximum value of any entry in the pheromone matrix is a, since an entry can receive a double update for one evaporation phase. Pheromone values above a are however very rare, since it is

there are no restrictions on the values that entries in the pheromone matrix could grow to. These systems are at risk of early convergence if the pheromone build-up on some entries reach very high values compared to the rest of the matrix. This risk is eliminated to a large degree by the proposed set-up. Any rational number can be used for a, as long as the same number is used consistently throughout the run of the algorithm. For this dissertation a value of $a = 1$ was used in all calculations.

5.5.5 Direct evaluation and summation evaluation

When ants employ equation 5.3.2 during the activity selection process, they are said to be following a local or direct evaluation strategy. This stems from the fact that ants only inspect the pheromone value τ_{ij} when evaluating the probability of scheduling activity j in position i. An alternative set-up, known as summation evaluation, has however also been proposed by Merkle et al. (2002). Summation evaluation gives ants a more global overview of the pheromone values by allowing them to evaluate $\sum_{k=1}^{i} \tau_{kj}$ instead of only τ_{ij} when calculating the likelihood of scheduling activity j in position i. Equation 5.5.5 shows the probability distribution associated with the selection of activities when ants follow a summation evaluation strategy:

$$p'_{ij} = \frac{(\sum_{k=1}^{i}[\gamma^{i-k} \cdot \tau_{kj}])^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{h \in \xi}((\sum_{k=1}^{i}[\gamma^{i-k} \cdot \tau_{kh}])^{\alpha} \cdot [\eta_{ih}]^{\beta})} \tag{5.5.5}$$

Where $\gamma > 0$ controls the relative influence of pheromone values corresponding to preceding positions. Summation evaluation has been successfully applied to the single machine total tardiness problem, where it is crucial for jobs not to be scheduled too late. For the RCPSP it is important that certain activities are not scheduled too late, since it could have dire implications on the deadline of the schedule. Summation evaluation is well suited for these situations. Resource feasible time slots are however irregular, and it is very possible that an activity can be scheduled in two different positions and still produce a good schedule, whereas all positions in between might have less than desirable results. For these situations, direct evaluation provides a better fit. With this in mind, Merkle et al. (2002) decided to make use of a hybrid approach where a combination of direct and summation evaluation is used to calculate probabilities. They still made use of equation 5.3.2 to calculate probabilities, but instead of using τ_{ij}, a modified value τ'_{ij} was introduced:

$$\tau'_{ij} = c \cdot x_i \cdot \tau_{ij} + (1 - c) \cdot y_i \cdot \sum_{k=1}^{i}(\gamma^{i-k} \cdot \tau_{kj}) \tag{5.5.6}$$

unlikely that a single entry would receive a double update for every single generation of the algorithm.

With $x_i = \sum_{h \in \xi} \sum_{k=1}^{i} (\gamma^{i-k} \cdot \tau_{kh})$ and $y_i = \sum_{h \in \xi} \tau_{ih}$.

Parameter value c controls the relative influence of direct evaluation and summation evaluation, with $c = 1$ leading to direct evaluation only, and $c = 0$ corresponding to pure summation evaluation. Its not always easy to predict what might happen if parameter values c and γ are adjusted. For this reason Merkle et al. (2002) experimented with various values and combinations for these two parameters to better understand the influence that they can have on the colony's performance. They found that the combination of $c = 0.5$ and $\gamma = 1$ produced the best results. If project managers want to adjust these values, then low values of γ and high values of c should be avoided, since these proved to deliver the worst results.

5.5.6 Pheromone evaporation rate

The pheromone evaporation rate controls how wide the colony of ants will search around good solutions. A low evaporation rate allows the colony to investigate a wide search area, but it might take a long time to converge on a really good solution. With a high evaporation rate, ants tend to concentrate around a specific area in the solution space. This might lead to ants getting stuck in local optima, resulting in premature convergence. Selecting an appropriate pheromone value is therefore crucial to the optimal functioning of the ACO. This selection process is however not trivial, and it is complicated by the fact that different problem instances will have a different response to the same pheromone value. Merkle et al. (2002) do not experiment with different pheromone values, but they simply start off with a pheromone value of $\rho = 0.025$ which is increased to $\rho = 0.075$ for the last 200 generations. The idea being that the colony should start with a low pheromone value to explore a wide area of the search space, which needs to be increased to a higher value towards the end of the run to ensure that the colony can converge in the allotted 850 generations. They do however note that a lower pheromone value can increase the solution quality of the colony if the algorithm run time is not a concern. This is also confirmed in an experimental study conducted by Dorigo and Stutzle (2001). Their experiments indicate that a lower pheromone evaporation rate produced superior solutions to higher values in almost all instances. It therefore seems that if time is not a factor, lower evaporation rates should be preferred. It is however important to give the colony sufficient time to reach convergence if these lower evaporation rates are used. Should the algorithm be terminated before convergence is reached, solution quality might be poor. The amount of time needed for convergence is however highly dependent on the problem complexity, and the pheromone evaporation rate. For complex problem instances a large number of generations of ants might be required before convergence is reached. Computation times might become impractical for these scenarios. Engineering projects can vary in size and complexity, and project managers will need guidance to select a pheromone evaporation

rate that can produce a high quality solution in a reasonable amount of time. An understanding of the relationship between pheromone evaporation, colony convergence and problem complexity is therefore required. Unfortunately academic literature on this topic is sparse to non-existent. An experimental study is therefore required to better understand these relationships. This experimental study will be the focus of the section that follows.

5.5.7 Experiments

The amount of time that project managers can invest in schedule optimisation is very much project specific. For large projects with a restricted budget, schedule optimisation can result in large cost savings, and both the client and the project manager would be more than willing to invest a significant amount of time on schedule optimisation. This might not be the case for smaller projects where the impact of schedule optimisation might be minimal. It therefore makes sense to let the project manager dictate how much time should be allotted to the runtime of the ACO. This time restriction can also be expressed in terms of the maximum number of generations that a colony can spawn. (This can be approximated by dividing the specified time limit by the runtime of one generation of ants.) Project managers are therefore faced with the task of determining a pheromone evaporation rate that will allow the colony to converge on a good solution within the allotted number of generations. The complexity of the problem instance under consideration will also have to be taken into consideration, since this could also influence the convergence behaviour of the colony. Academia provides almost no information as to how this selection process should be approached. The goal of this section is therefore to conduct an experiment that can give insight into the influence that pheromone evaporation and project complexity can have on the convergence behaviour of the colony. Since the ACO has a random component, the convergence behaviour of a colony of ants will never be 100% predictable. The end goal of this experiment is therefore not to find a hard rule that project managers must use to select a pheromone evaporation rate, but rather to provide a guideline that can be used as a good starting point when choosing the input parameters of the ACO.

5.5.7.1 Setup

Due to the extensive research that the RCPSP has enjoyed, several benchmark libraries have been developed for researchers to test their algorithms on. One of the more popular libraries, *PSPLIB* (available at http://www.om-db.wi.tum.de/psplib/), is of particular interest. Not only is this one of the most popular and largest project libraries on offer, but the researchers who developed the library Drexel et al. (1995) also devoted much attention to classify

the complexity of a resource-constrained project instance. Three parameters were identified as being key to measuring this complexity:

Network complexity Network complexity (NC) is an indication of the density of the technical precedence graph of a project. It is measured as the average number of non-redundant edges per node (activity) in the technical precedence graph. Interestingly, projects with a higher network complexity are generally easier to solve than projects with a lower network complexity (Alvarez-Valdes and Tamarit, 1989). Increasing the network complexity of a project leads to a more constrained environment, with less possible schedule combinations. The eligible sets generated by the SSGS will typically be smaller, leading to a smaller solution space, and also faster computation times.

Resource strength Resource strength (RS) is a measure of the resource scarcity of a project. It is meant to express the relationship between the resource demands of activities and the resource availability of a project. Resource strength was first introduced by Cooper (1976). Drexel et al. (1995) provided several enhancements to the way in which resource strength should be measured, and they also normalised RS over the interval [0, 1]. With $RS = 0$ corresponding to projects with just enough resources to generate a feasible schedule, and $RS = 1$ corresponding to the unconstrained case. Projects with a low resource strength are therefore harder to solve than projects with a high resource strength. The resource strength of a project is said to be type dependent, since different resource types will not necessarily have the same resource strength.

Resource factor The resource factor (RF) of a project reflects the average diversity of the resource usage of project activities. In other words, how many different resource types are included in an activity's resource requirements. The resource factor was first introduced by Pascoe (1966), and only slightly modified by Drexel et al. (1995). RF is also normalised over the interval [0, 1]. If $RF = 1$, then every activity in the project requires at least one resource of each resource type. If $RF = 0$, then all activities in the project have zero resource requirements. Projects with a high resource factor are generally harder to solve than projects with a low resource factor.

Besides these three factors, the number of activities obviously also influences the complexity of a project. Drexel et al. (1995) developed a random network generator called *ProGen*, capable of generating fictitious projects of any size, network complexity, resource strength and resource factor. They dedicated significant effort to ensure that the projects produced by *ProGen* are realistic, which arguably led to *ProGen* being one of the most popular random network generators around. All of the problem instances in the *PSPLIB* were

also generated using *ProGen*. *ProGen* is therefore a logical choice for generating project instances for experimentation purposes. It offers a controlled environment where the relationship between pheromone evaporation, project complexity and ant colony convergence can be studied. *ProGen* can be used to generate a fixed number of problem instances conforming to predetermined size and complexity settings. These problems can then be scheduled with a variety of different pheromone evaporation rates, to see if there is any correlation between convergence speed, project complexity and pheromone evaporation.

PSPLIB already contains a large number of project instances varying in complexity and size. The most complex problem sets in the library consist of 120 activities each. Informal surveys indicate that the number of activities in the design phase of civil engineering projects typically range between 60 - 1000. Several of these problem instances are therefore useful, but it was necessary to generate additional problem sets with a higher activity count.

In the end, a total of 100 projects were selected for experimentation purposes. Ten problem sets consisting of ten projects each were used. All projects in a problem set were generated with the same complexity parameters. These parameters are shown in Table 5.1. Note that the project sets are listed in order of increasing size and complexity.

Table 5.1: Complexity Parameters of Problem Sets

Problem Set	j	NC	RF	RS
1	60	2.1	1.00	0.2
2	60	1.5	1.00	0.2
3	120	2.1	0.5	0.3
4	120	1.5	1.00	0.1
5	250	2.1	0.5	0.3
6	250	1.5	1.00	0.1
7	500	2.1	0.5	0.3
8	500	1.5	1.00	0.1
9	1000	2.1	0.5	0.3
10	1000	1.5	1.00	0.1

Problem sets 1-4 were acquired from the *PSPLIB*, whilst problem sets 4-10 were newly generated with *ProGen*. Problems sets 1-4 correspond to the *PSPLIB* problem sets J60-45, J60-13, J120-48 and J120-16 respectively. Three different solutions were generated for each test project with the ACO described in section 5.3.3. A different pheromone value was used for each run. Slightly higher pheromone values were used for the bigger problem instances, since they were expected to take too long to converge with small pheromone values. The pheromone values used for each problem set is shown in Table 5.2:

Table 5.2: Pheromone Values for Respective Runs

Problem Set	$\rho 1$	$\rho 2$	$\rho 3$
1	0.01	0.025	0.05
2	0.01	0.025	0.05
3	0.025	0.05	0.1
4	0.025	0.05	0.1
5	0.05	0.1	0.15
6	0.05	0.1	0.15
7	0.1	0.15	0.2
8	0.1	0.15	0.2
9	0.1	0.2	0.3
10	0.1	0.2	0.3

In order to minimise the impact of the random component of the ACO, this whole process was repeated three times. This gives a total of $3 \times 3 \times 100 = 900$ ant colonies that can be analysed. Each colony was allowed to run until convergence was reached. The number of colonies required to reach convergence was noted, along with the run time and schedule makespan. All tests were written in the Java programming language, and executed on a personal computer with a 3.10 GHz Intel quad core processor.

5.5.7.2 Results

The results for each problem set are summarised in tables 5.3 - 5.12. Several properties are noted for each problem set (PS):

***g*:** The number of generations needed for convergence was noted for each of the different pheromone values. The numbers shown are averaged over the three runs that each problem set was subject to.

t: The average runtime until convergence is shown for every pheromone value and problem set combination. Values are presented in seconds.

***dev*:** For each pheromone value, the best solution (shortest makespan) obtained in the three runs was noted. The best solutions obtained with each of the pheromone values were then compared to each other. The deviation from the minimum of these values were noted for each pheromone value. The results are shown in the *dev* column of the tables. Values are presented as percentages. A value of 0% is therefore an indication of the pheromone value that achieved the best solution for a specific problem set.

When analysing the results shown in the tables below, it is important to compare problem sets that only differ in regard to one parameter. In other

words, problem set 1 can be compared to problem set 2, since they only differ in regards to their network complexity. It is therefore possible to say that problem set 2 is more complex than problem set 1, since all of their other complexity parameters are equal. The same can however not be said for problem set 4 and 5 for example. Even though problem set 5 has more activities than problem set 4, problem set 4 has lower NC and RS values, as well as a higher RF value. It is therefore not clear which problem set has a higher total complexity. Problem set 4 and 6 could however be compared, since problem set 6 definitely has a higher complexity than problem set 4.

Table 5.3: Convergence Results: Problem Set 1

	$\rho = 0.01$			$\rho = 0.025$			$\rho = 0.05$		
PS	*g*	*t*	*dev*	*g*	*t*	*dev*	*g*	*t*	*dev*
1-01	5137.67	39.31	0.0	1638.0	13.63	1.01	288.0	2.51	5.05
1-02	6167.0	52.88	0.68	1410.33	13.89	0.0	374.33	3.92	3.38
1-03	7500.33	69.88	0.67	1348.33	13.3	0.0	493.33	5.02	0.67
1-04	5992.67	47.14	0.0	1073.67	8.97	0.0	368.33	3.19	0.89
1-05	7679.67	63.63	0.0	1617.67	14.47	0.92	382.33	3.5	1.83
1-06	7757.33	70.78	0.0	1446.67	14.0	0.0	419.33	4.16	1.34
1-07	6198.33	47.36	0.0	1491.33	13.44	0.0	340.0	3.28	1.6
1-08	5316.33	45.25	1.52	993.67	9.36	0.76	422.0	3.99	0.0
1-09	6105.33	51.49	2.33	1103.0	9.81	0.0	376.67	3.52	0.0
1-10	8092.0	65.49	1.67	1209.0	10.52	0.0	366.67	3.32	1.67
Avg:	**6594.67**	**55.32**	**0.69**	**1333.17**	**12.14**	**0.27**	**383.1**	**3.64**	**1.64**

Table 5.4: Convergence Results: Problem Set 2

	$\rho = 0.01$			$\rho = 0.025$			$\rho = 0.05$		
PS	*g*	*t*	*dev*	*g*	*t*	*dev*	*g*	*t*	*dev*
2-01	12467.0	132.1	0.0	2158.67	24.94	0.85	662.67	7.88	1.69
2-02	7569.0	71.85	0.91	1945.0	21.45	0.0	529.67	6.19	2.73
2-03	12627.33	119.12	0.0	2395.67	25.39	1.1	787.33	8.96	1.1
2-04	8431.0	80.92	0.0	1811.0	19.95	0.0	428.67	4.85	0.94
2-05	8973.67	88.27	1.0	2189.0	22.45	0.0	460.67	4.88	1.0
2-06	12410.67	122.43	0.0	2047.67	22.06	1.03	380.67	4.3	3.09
2-07	10357.67	99.52	2.22	2194.33	24.0	0.0	650.67	7.44	0.0
2-08	9511.33	85.98	0.0	2065.67	23.18	1.6	489.33	5.77	1.6
2-09	7337.0	59.18	0.0	1561.0	13.79	0.95	341.67	3.19	0.95
2-10	9474.0	89.96	0.0	1937.0	21.39	0.83	626.33	7.4	0.83
Avg:	**9915.87**	**94.93**	**0.41**	**2030.5**	**21.86**	**0.64**	**535.77**	**6.09**	**1.39**

Table 5.5: Convergence Results: Problem Set 3

	$\rho = 0.025$			$\rho = 0.05$			$\rho = 0.1$		
PS	*g*	*t*	*dev*	*g*	*t*	*dev*	*g*	*t*	*dev*
3-01	4871.67	111.36	0.0	1072.33	26.41	0.0	282.0	7.16	0.0
3-02	4989.33	121.58	0.86	1309.33	34.68	0.0	321.67	8.64	0.86
3-03	5300.33	127.8	0.0	1096.0	27.09	0.0	248.33	6.33	2.54
3-04	6256.67	151.54	0.0	1225.0	30.59	0.0	318.33	8.08	2.22
3-05	4548.33	102.72	0.0	1084.67	25.81	0.87	226.33	5.7	0.87
3-06	7447.33	175.62	0.0	871.33	21.77	0.0	267.0	6.99	0.0
3-07	5025.0	119.57	0.0	1113.0	27.68	1.8	341.33	8.66	1.8
3-08	3866.33	83.07	0.0	687.33	16.11	0.83	233.67	5.75	0.83
3-09	6357.33	147.35	0.0	1093.33	27.57	0.0	238.0	6.32	1.69
3-10	3657.0	84.04	0.0	913.33	21.96	0.86	281.33	7.02	0.86
Avg:	**5231.93**	**122.46**	**0.09**	**1046.57**	**25.97**	**0.44**	**275.8**	**7.07**	**1.17**

Table 5.6: Convergence Results: Problem Set 4

	$\rho = 0.025$			$\rho = 0.05$			$\rho = 0.1$		
PS	*g*	*t*	*dev*	*g*	*t*	*dev*	*g*	*t*	*dev*
4-01	13722.33	467.34	0.0	1989.33	78.73	1.44	549.0	22.19	3.85
4-02	10396.33	375.77	0.41	2091.67	86.81	0.0	473.67	20.37	1.63
4-03	11856.33	463.21	0.0	2142.33	92.6	0.8	440.0	19.8	2.81
4-04	13043.0	457.17	0.94	2754.33	106.99	0.0	645.67	25.31	1.89
4-05	8725.33	300.65	0.0	2046.67	75.22	0.0	569.67	21.09	2.35
4-06	8675.33	335.39	0.0	2558.67	106.41	1.86	642.67	28.03	4.19
4-07	9344.67	355.98	0.0	2220.67	92.91	2.06	521.0	22.44	2.58
4-08	9526.0	364.46	0.0	2147.0	87.45	0.49	510.0	21.54	2.43
4-09	9975.67	409.5	0.0	2031.33	88.28	0.93	581.33	26.26	2.31
4-10	10283.33	379.74	0.44	1856.67	77.64	0.0	509.33	21.85	1.76
Avg:	**10554.83**	**390.92**	**0.18**	**2183.87**	**89.3**	**0.76**	**544.23**	**22.89**	**2.58**

Table 5.7: Convergence Results: Problem Set 5

	$\rho = 0.05$			$\rho = 0.1$			$\rho = 0.15$		
PS	*g*	*t*	*dev*	*g*	*t*	*dev*	*g*	*t*	*dev*
5-01	2368.67	306.28	0.0	1192.0	158.25	0.0	414.33	55.63	0.74
5-02	3399.0	444.16	0.0	880.0	118.84	0.0	342.0	46.72	0.66
5-03	2843.67	377.82	0.65	681.0	92.38	0.0	283.0	39.01	0.0
5-04	2099.33	300.19	0.0	681.33	104.4	0.0	356.0	54.34	0.0
5-05	2174.67	295.83	0.78	967.0	136.59	0.0	397.67	57.45	0.0
5-06	975.67	138.65	0.0	308.67	46.86	0.0	257.33	38.05	0.0
5-07	4178.33	524.14	0.0	771.0	101.99	0.71	326.67	44.03	0.0
5-08	2122.33	283.06	1.33	1286.33	182.53	0.0	351.0	49.94	0.0
5-09	3895.0	491.85	0.62	1010.67	132.31	0.0	401.67	53.78	1.23
5-10	4543.33	626.01	0.8	858.33	120.14	0.0	497.33	69.62	0.8
Avg:	**2860.0**	**378.8**	**0.42**	**863.63**	**119.43**	**0.07**	**362.7**	**50.86**	**0.34**

Table 5.8: Convergence Results: Problem Set 6

	$\rho = 0.05$			$\rho = 0.1$			$\rho = 0.15$		
PS	*g*	*t*	*dev*	*g*	*t*	*dev*	*g*	*t*	*dev*
6-01	11576.67	2353.34	0.0	1944.0	413.14	1.52	527.0	114.2	1.83
6-02	9607.67	2166.42	0.0	1581.67	370.03	0.33	523.0	125.06	2.63
6-03	9072.67	1922.58	0.0	1731.67	379.9	0.0	584.67	129.62	1.08
6-04	7743.33	1702.13	0.0	1528.67	345.07	0.68	607.33	138.61	1.71
6-05	9764.33	2085.29	0.0	1375.67	298.7	0.99	531.33	118.86	1.66
6-06	9711.0	2100.08	0.0	1447.0	317.94	2.05	532.67	121.07	1.71
6-07	9734.0	1950.04	0.0	1680.67	339.31	2.03	485.0	98.68	2.7
6-08	8231.67	1615.33	0.0	1486.67	295.18	0.71	507.33	101.89	2.47
6-09	8555.0	1703.62	0.0	1572.33	325.82	0.82	557.0	117.11	1.65
6-10	9589.67	2099.61	0.0	1383.67	311.62	1.06	543.67	124.47	2.11
Avg:	**9358.6**	**1969.84**	**0.0**	**1573.2**	**339.67**	**1.02**	**539.9**	**118.96**	**1.96**

Table 5.9: Convergence Results: Problem Set 7

	$\rho = 0.1$			$\rho = 0.15$			$\rho = 0.2$		
PS	*g*	*t*	*dev*	*g*	*t*	*dev*	*g*	*t*	*dev*
7-01	422.67	360.87	0.61	224.67	194.98	0.0	121.33	109.12	0.61
7-02	906.67	741.29	0.58	377.33	310.73	0.0	216.67	179.11	0.0
7-03	338.67	276.1	0.0	224.67	189.94	0.0	221.33	185.43	0.0
7-04	431.0	343.17	0.0	283.67	233.91	0.56	183.33	151.01	0.56
7-05	507.0	431.79	0.0	274.0	242.13	0.0	134.67	119.02	0.59
7-06	351.33	273.82	0.59	224.33	178.09	0.59	213.67	169.53	0.0
7-07	437.67	374.33	0.0	297.0	256.86	0.0	223.33	192.75	0.66
7-08	472.67	405.48	0.0	699.33	618.21	0.51	347.33	305.86	0.51
7-09	921.67	821.24	0.5	622.0	567.95	0.5	332.67	296.2	0.0
7-10	1072.67	1002.58	0.0	268.0	254.19	0.0	198.33	192.2	0.0
Avg:	**586.2**	**503.07**	**0.23**	**349.5**	**304.7**	**0.22**	**219.27**	**190.02**	**0.29**

Table 5.10: Convergence Results: Problem Set 8

	$\rho = 0.1$			$\rho = 0.15$			$\rho = 0.2$		
PS	*g*	*t*	*dev*	*g*	*t*	*dev*	*g*	*t*	*dev*
8-01	1341.33	1523.84	0.0	939.33	1074.95	0.33	351.0	405.28	0.65
8-02	1170.0	1430.82	0.3	993.67	1235.17	0.0	491.67	598.07	0.59
8-03	4002.33	5077.55	0.0	1056.67	1313.22	0.57	538.67	677.23	1.14
8-04	2961.0	3548.86	0.0	936.67	1108.67	0.0	421.0	497.31	0.8
8-05	4075.67	5097.41	0.26	1189.67	1481.82	0.0	491.0	615.31	1.03
8-06	2631.67	3164.64	0.84	1029.33	1243.26	0.0	456.67	554.39	0.84
8-07	924.0	1188.45	0.37	624.33	807.36	0.0	289.0	374.12	0.75
8-08	3590.0	4411.36	0.0	1036.0	1282.56	0.0	374.0	464.69	0.57
8-09	1863.33	2303.95	0.0	904.33	1118.43	0.0	441.67	551.32	0.0
8-10	2098.33	2571.55	0.0	923.67	1153.29	0.0	459.67	569.29	0.58
Avg:	**2465.77**	**3031.84**	**0.18**	**963.37**	**1181.87**	**0.09**	**431.43**	**530.7**	**0.7**

Table 5.11: Convergence Results: Problem Set 9

	$\rho = 0.1$			$\rho = 0.2$			$\rho = 0.3$		
PS	*g*	*t*	*dev*	*g*	*t*	*dev*	*g*	*t*	*dev*
9-01	373.67	2034.52	0.5	151.0	832.22	0.0	98.0	582.57	1.01
9-02	370.0	1985.77	0.0	167.33	927.4	0.47	88.0	523.22	0.93
9-03	324.67	1741.42	0.49	219.67	1235.12	0.0	90.33	548.29	0.99
9-04	272.33	1582.95	0.0	89.67	554.16	0.0	122.0	767.97	0.0
9-05	404.67	1962.7	0.0	143.0	738.19	0.0	80.33	439.89	0.51
9-06	353.0	1835.76	0.0	146.67	787.37	0.0	141.0	799.05	0.0
9-07	312.33	1728.19	0.0	257.0	1465.28	0.52	159.67	913.89	0.0
9-08	655.67	3247.76	0.0	208.33	1060.34	0.0	64.67	361.11	1.08
9-09	291.67	1701.3	0.54	166.67	1000.59	0.0	100.67	650.05	0.54
9-10	357.0	2049.56	0.0	101.67	614.18	0.0	74.33	477.5	0.56
Avg:	**371.5**	**1986.99**	**0.15**	**165.1**	**921.48**	**0.1**	**101.9**	**606.35**	**0.56**

Table 5.12: Convergence Results: Problem Set 10

	$\rho = 0.1$			$\rho = 0.2$			$\rho = 0.3$		
PS	*g*	*t*	*dev*	*g*	*t*	*dev*	*g*	*t*	*dev*
10-01	1036.67	8376.5	0.0	362.0	3033.52	0.49	242.33	2019.07	0.74
10-02	1205.33	10022.65	0.0	330.0	2788.9	0.23	197.0	1692.41	0.45
10-03	826.67	6168.69	0.0	332.33	2538.12	0.0	151.33	1198.19	0.26
10-04	1102.33	8855.8	0.0	369.67	3013.7	0.29	145.0	1237.47	0.57
10-05	784.0	6690.9	0.0	235.67	2047.79	0.28	191.0	1698.21	0.28
10-06	740.67	5858.59	0.0	306.0	2476.36	0.53	178.0	1464.8	1.06
10-07	915.0	7239.7	0.0	493.67	3953.63	0.48	254.33	2077.79	0.48
10-08	771.33	5826.51	0.0	257.33	1996.94	0.56	173.67	1371.09	0.83
10-09	623.33	4881.95	0.0	212.0	1744.86	0.32	130.33	1099.7	0.95
10-10	671.33	5454.63	0.0	346.67	2859.72	0.0	254.0	2129.54	0.0
Avg:	**867.67**	**6937.59**	**0.0**	**324.53**	**2645.35**	**0.32**	**191.7**	**1598.83**	**0.56**

5.5.7.3 Remarks and Observations

Several interesting observations can be made when analysing the data of the convergence experiment. The most noteworthy ones are discussed in this section.

Inverse relationship between pheromone value and generations required for convergence If the complexity of a project is fixed, then an inverse relationship exists between a colony's pheromone evaporation rate and the number of generations required for convergence to be reached. Increasing the pheromone value of a colony, decreases the number of generations required for convergence and visa versa. This holds true for every single problem set. This was however expected, and also correlates with previous research.

Low pheromone values do not always guarantee the best solution Previous research has indicated that lower pheromone values should be preferred over higher values, since they will produce superior solutions if runtime is not a factor. The data of this experiment has however shown that this is not always true. $\rho 2$ produced superior results to $\rho 1$ for several problem sets. The performance difference was however small, and $\rho 1$ never produced a schedule that exceeded the makespan of the best solution by more than 1%. $\rho 3$ consistently offered the worst performance, but also did not deviate from the best solution by more than 3%. These differences might become more pronounced if pheromone values differed by a larger margin. Overall, the lower pheromone values did however perform the best, and as a rule of thumb low pheromone values should be preferred to high pheromone values. The performance gain in solution quality is however not always worth the additional run time.

Runtime becomes significant for large problem instances For the larger, more complex problem instances the algorithm needed a significant amount of time to converge. Problem set 10 is the most complex problem set investigated, and required roughly 8 seconds to complete one generation. Two operations performed by the ACO are computationally expensive: Finding a resource feasible slot for each activity, and pheromone evaporation. Increasing the number of activities in a project will also increase the complexity of these two operations. Algorithm runtime is therefore strongly correlated to the number of activities in a project. Project managers should keep these run times in mind when setting up the input parameters of an ACO, since it could possibly take several hours to reach convergence for large projects.

Relationship between pheromone evaporation, project complexity and convergence The results of these experiments suggest that a fixed relationship does indeed exist between pheromone evaporation, project complex-

ity, and the number of generations required to reach convergence. Inspection revealed that this relationship can be approximated by

$$g = \frac{1}{\rho^x} \tag{5.5.7}$$

where x is a function of j, NC, RS and RF. To illustrate this relationship, the x value for each of the problem sets analysed was calculated. The results are shown in table 5.13.

Table 5.13: x-values of problem sets

	x-values			
Problem set	ρ 1	ρ 2	ρ 3	Ave.
1	1.91	1.95	1.98	1.95
2	2	2.06	2.09	2.05
3	2.32	2.32	2.44	2.36
4	2.51	2.56	2.73	2.6
5	2.66	2.93	3.1	2.9
6	3.05	3.19	3.3	3.18
7	2.77	3.08	3.34	3.06
8	3.4	3.62	3.77	3.6
9	2.57	3.17	3.84	3.19
10	2.94	3.59	4.37	3.63

The x values of problem sets with the same complexity tend to be constant, with minor deviations from the average. This relationship does however break down for high pheromone values, where deviations become more accentuated (refer to problem sets 9 and 10). Premature optimisation is much more likely to occur with higher pheromone values, and is most likely the cause of the slightly erratic behaviour observed in the larger problem sets. Further research might refine the relationship presented in equation 5.5.7, but the random component of the ACO will always inhibit this relationship from being captured with 100% accuracy for all problem sets. Nevertheless, equation 5.5.7 still provides a good heuristic for selecting an appropriate initial pheromone value for an optimisation process. If the total project complexity is known, and a corresponding x value is available, then a project manager could use equation 5.5.7 to calculate a pheromone value that would let the colony converge in the desired number of generations. Unfortunately there exists no formula to calculate an x value based on the project complexity at present. It seems like more complex problem instances require higher x-values, but the matter still requires further research. If no fixed relationship can be found, then it might be necessary to calculate these x values empirically by generating an even larger test case than the one provided in this dissertation. This process could also be done over time by crowd-sourcing results from project managers

using the tool. For the time being, project managers will have to base their x values off the results of this experiment, by means of basic interpolation. This should already provide a very good starting point for finding an appropriate pheromone evaporation rate.

5.6 Implementation details - *ProBaSE*

For the ACO to become practically useful, a software implementation is required that can be used by project managers. At the time of writing, no commercial scheduling packages have incorporated an ACO in their scheduling engines. Without the appropriate software tools, the ACO can never be expected to gain traction in the civil engineering industry. It was therefore important to develop a software package that not only implements the ACO, but also implements the complete BaSE framework. This goal was achieved with the development of *ProBaSE*, which is a prototype Java implementation of the BaSE framework. *ProBaSE* along with source code is provided on the supplementary material CD accompanying this dissertation. This section will focus on the ACO capabilities of *ProBaSE*, and will illustrate how they can facilitate project managers during the resource-constrained scheduling phase.

5.6.1 A graphical user interface

ProBaSE provides the project manager with a graphical user interface from which all scheduling related activities can be initiated. Some of the core features pertaining to resource-constrained scheduling include:

- Importing resource-constrained projects.
- Analysing project complexity.
- Adjusting the ACO input parameters.
- Analysing the behaviour of an ant colony.
- Viewing an optimised schedule.
- Exporting the optimised schedule.

Figure 5.3 shows the main screen from where these activities would be launched.

5.6.2 Preparing for schedule optimisation

Two steps are required from the project manager before schedule optimisation can commence. A resource-constrained project needs to be imported, and the input parameters of the ACO need to be fixed.

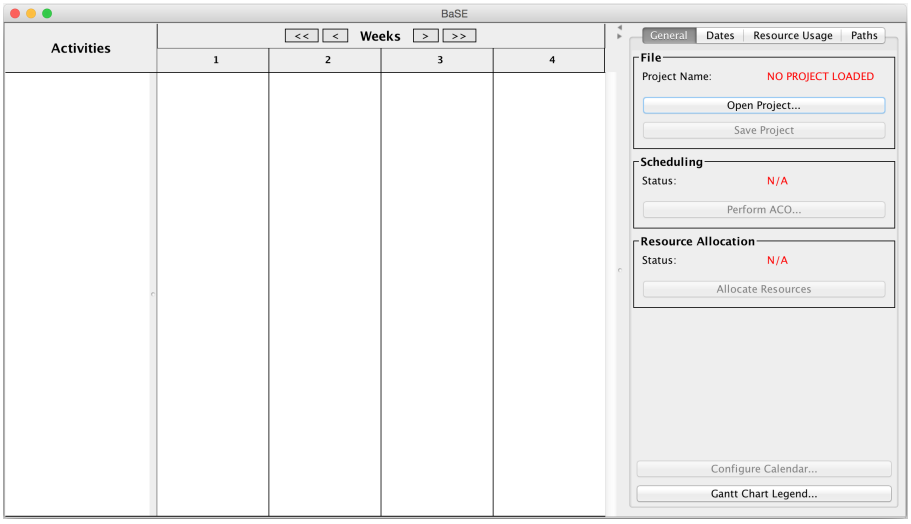


Figure 5.3: *ProBaSE* - Main View

Importing the project is trivial, and simply requires the project manager to open an appropriate input file from the main window. The format of this input file will be the subject of discussion in section 5.6.6.

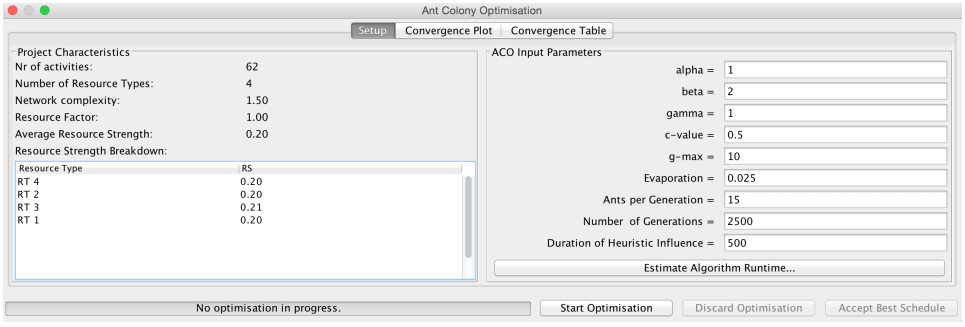


Figure 5.4: *ProBaSE* - ACO window

With a resource-constrained project loaded, project managers can navigate to the ACO window shown in figure 5.4 to launch the scheduling process. Here project managers can define the values of input parameters, and initiate the ACO. To assist with the selection of appropriate parameter values, the most important properties of the imported project will automatically be calculated and displayed. This will include the number of activities, the number of resource types, the network complexity, the resource scarcity and the resource strength of the project. The guidelines provided in section 5.5 should then be used to select appropriate input parameters. Most of these fields will already be populated with good starting values by default, but parameters such as pheromone evaporation and the number of generations might still require adjustment depending on the complexity of the project, and the amount of

time available for optimisation. Functionality is provided to estimate the runtime of the ACO based off the input parameters provided. This is achieved by simply performing a short dry run, and estimating the time of one generation of ants. This information should help a project manager decide on the maximum generations of ants that the colony can be allowed to spawn. A suitable pheromone value can then be calculated using equation 5.5.7. x values should be interpolated from the values provided in section 5.5.7.3, or it can be based off historical data.

5.6.3 Analysing the behaviour of the colony

As stated in section 5.4, it must be possible for the project manager to evaluate the performance and behaviour of the algorithm. To facilitate this process, a graph is presented to the user that tracks the change in solution quality over time. On the x-axis, the number of ant generations analysed is displayed. On the y-axis, the makespan of the best solution found in a specific generation is shown. By analysing this graph, project managers can get an idea of the convergence behaviour of the colony. Ideally the colony would only converge on the best solution towards the end of the run. If premature convergence is evident, then a lower pheromone value should be used in order to explore a larger region of the solution space. An example of this is shown in figure 5.5. If the colony does not display convergence behaviour, then a larger pheromone

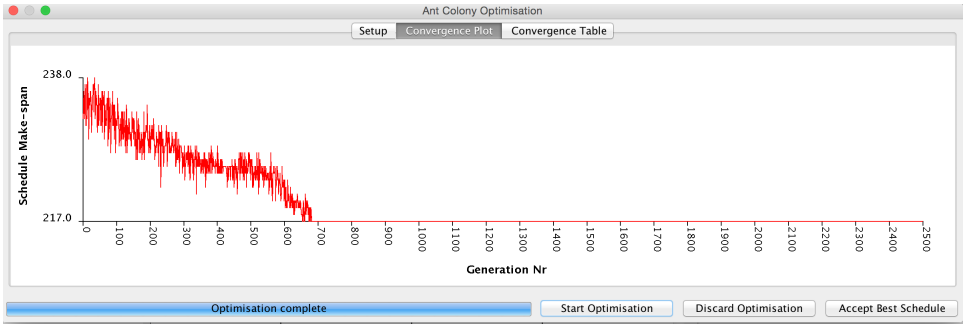


Figure 5.5: Premature convergence

value should be used or the colony should be allowed to spawn more generations of ants. Such behaviour is displayed in figure 5.6. Functionality is provided to pause the ACO at any stage during a run. Convergence behaviour can therefore be monitored at intermediate stages, without having to wait for a run to finish. If the progress of the colony is not satisfactory, the algorithm can be reset with more suitable parameter values. This should hopefully speed up the process of finding a good solution. Figure 5.7 shows the results of a colony that displayed good convergence behaviour in the allotted amount of time. The data for these plots are also presented to the user in table form, so

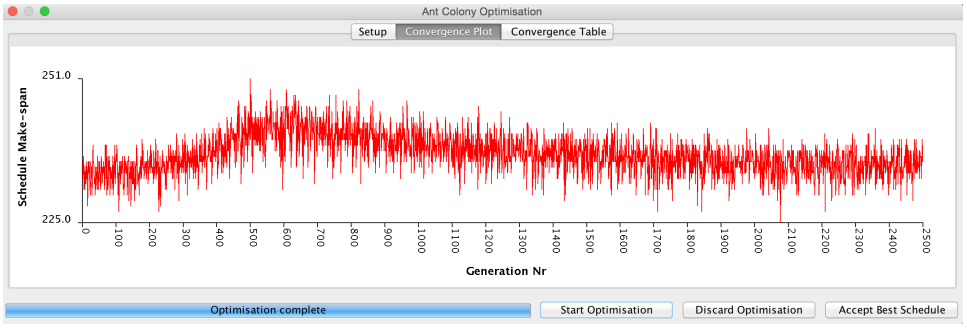


Figure 5.6: Non-convergent behaviour

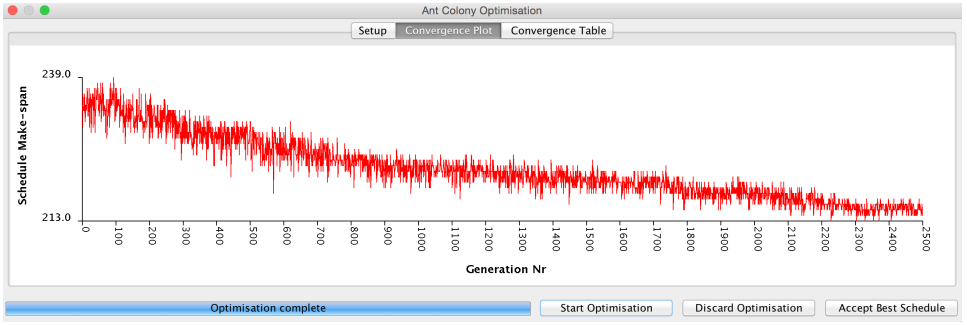


Figure 5.7: Convergent behaviour

project managers could analyse the behaviour of a colony with any statistical tool of their liking.

5.6.4 Configuring the project calendar

The results produced by the ACO scheduling algorithm are based on time increments, and do not factor in any calendar related time information. Projects executed in the real world are however very much calendar based, and the schedule needs to reflect information such as the project start- and end-date, as well as project off days. Functionality is therefore provided for the project manager to specify this information. The project start date can be specified, as well as the durations of a standard work week. Calendar days that do not conform to the specifications of a standard work week can also be configured. All of this information can be specified in the project input file, or it can be done graphically using the calendar tool shown in figure 5.8. The schedule produced by the ACO will then automatically be mapped onto this calendar, to provide project managers with realistic activity start- and end-dates. Refer to the source code of *ProBaSE* to see exactly how this is achieved. Since *ProBaSE* is meant to be used during the engineering-planning phase, detailed information regarding the shifts of individual resource would not be available. Resource calendars therefore need not be specified at this stage of project planning.

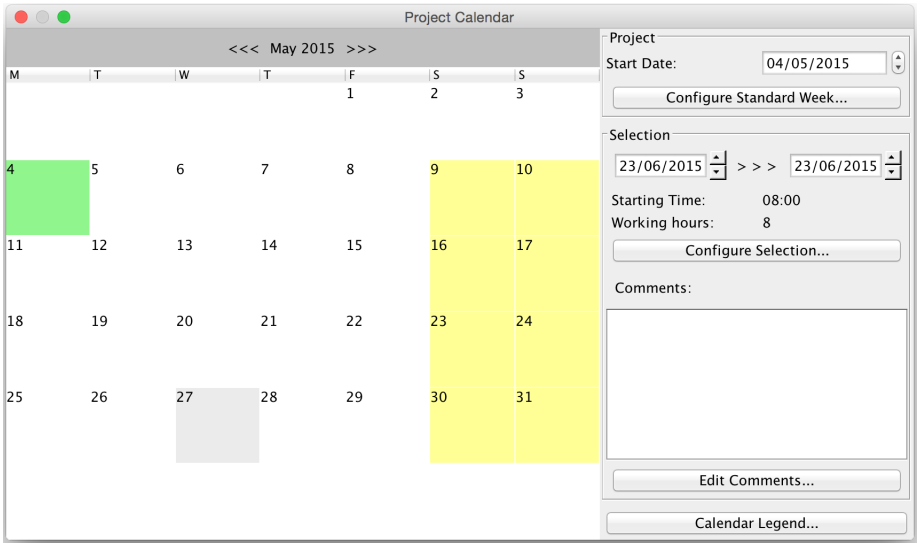


Figure 5.8: *ProBaSE* - Project Calendar

5.6.5 Viewing the optimised schedule

ProBaSE presents the optimised schedule to the user in the form of a Gantt chart. Gantt charts have been widely used by project managers for many years now, since they offer such an effective way to visualise a project schedule. Important project information such as activity start- and end-dates, precedence relations and critical paths can easily be presented on a Gantt chart. The Gantt chart viewer of *ProBaSE* is shown in figure 5.9. Chapter 6 will explain the functionality of this viewer in more detail.

5.6.6 XML based project files

Most commercial project management packages in use offer much more functionality than just scheduling. The goal of *ProBaSE* is not to replace these project management packages, but rather to facilitate with one of the more complicated phases, namely resource-constrained scheduling. It is therefore important that information can be easily exchanged between *ProBaSE* and commercial packages. With this in mind, it was decided to make use of the well known XML format for the storage of project files. The minimum input file will consist of the project specifications, and it will include activity names, activity durations, a project calendar, resource constraints, resource requirements and all activity dependencies. *ProBaSE* can then automatically append all of the scheduling information that it generates to this file. This will include scheduled start and end dates, as well as resource allocations. There are several advantages to storing these files in this format:

- XML is an industry accepted file format for the exchange of information. Most software developers are familiar with it, and it would be straight

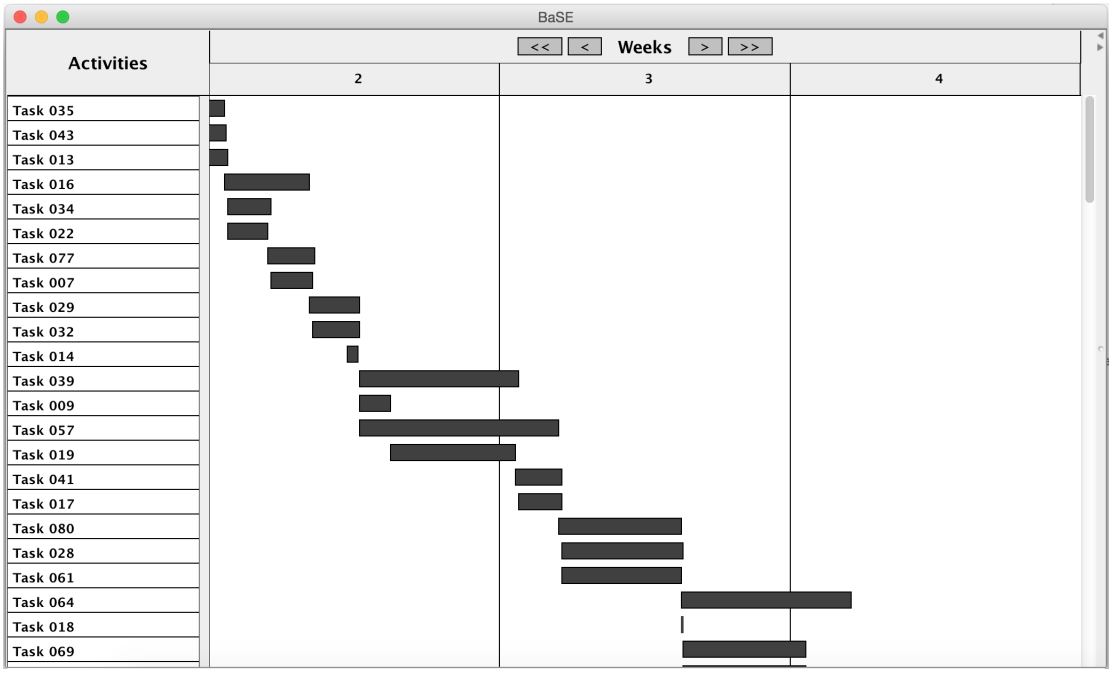


Figure 5.9: Gantt-chart viewer

forward to export and import information to and from an XML file. Commercial software vendors could easily develop such bridges if they want to use the sophisticated scheduling engine of *ProBaSE*.

- XML is human-readable. If no bridge is available between a commercial software package and *ProBaSE*, then the conversion process could easily be done by a human. Granted it will take more time, but it allows project managers to use any project management package of their choice.

The input and output files for an example project are given on the CD accompanying this dissertation.

5.7 Chapter summary and recommendations

This chapter investigated a solution for the resource-constrained scheduling phase of the BaSE framework. An ant colony optimisation was proposed as a viable solution to this problem. Not only has this optimisation proved to be one of the most effective meta-heuristics available for the RCPSP, but it also has the advantage of being very understandable and time-effective. This is important for the engineering industry where project managers do not necessarily have a background in optimisation and where problem sizes can become cumbersome.

The ACO finds no implementation in commercial software, and this chapter attempted to take the first steps toward making the ACO practical and usable

for the civil engineering industry. Two areas were identified as being key to this process: There has to be more clarity and guidance regarding the selection of good parameter values for the ACO, and a usable software implementation of the algorithm has to be provided.

Although Merkle et al. (2002) did experiment with different parameter values for the ACO, their experiments were limited to projects containing 120 activities. Engineering projects can vary in size and complexity, and parameter values often times will have to be tailored accordingly. One specific aspect that was left unexplored by Merkle et al. (2002), was the relationship between pheromone evaporation, problem complexity, and the convergence speed of a colony. Pheromone evaporation is one of the most important mechanisms to control the search behaviour of a colony, and for large problem instances it is crucial that an appropriate value is chosen. Experiments were therefore set up to gain a deeper understanding of this relationship. The results of these experiments seem to suggest that a fixed relationship does in fact exist between these parameters, and that this relationship can be approximated with the equation $g = \frac{1}{\rho^x}$, with x being a function of the project complexity. Unfortunately it is not clear how this x value can be calculated based on the project complexity alone, and further research is required into the matter. Options such as crowd-sourcing should be explored, where *ProBaSE* could automatically build up a database off x values, by tracking the convergence results of projects scheduled by users. This process will however require a significant amount of time and users before it can truly be effective. It will also require the consent of the companies that use the tool. The x values derived in this dissertation can however be used for the time being, and should provide a good starting point for most engineering projects.

An implementation of the ACO was also developed, complete with a graphical user interface. The implementation, named *ProBaSE*, offers a platform from which all schedule related activities can be performed. *ProBaSE* makes it easy to view the project complexity, experiment with different parameter values, analyse the behaviour of a colony and view optimised schedules. It accepts XML input files, and can export schedules in the same format. These file formats allow *ProBaSE* to be used in conjunction with other project management tools. Some tooling is still required to export and import these XML files into the more popular project management packages. Most of the established project management tools, such as MS Project, do however offer open API's, so the IT staff of an engineering company should not have too much trouble achieving this. The exporting/importing process could also be done manually if no software bridges are available.

Chapter 6

A heuristic approach for maximising slack in resource-constrained schedules

6.1 Introduction

In the previous chapter, a complete solution was developed for the first phase of BaSE. The schedules that this phase will produce conform to two of the four characteristics of high quality baseline schedules as defined in Chapter 2:

1. They will be based on complete and accurate input data. The ACO incorporates all of the most important constraints that are present during the engineering-planning phase. It accounts for both resource constraints and technical precedence constraints, and should produce accurate estimates if realistic activity durations are provided by project managers.

2. The schedules will be optimised. The ACO offers very sophisticated optimisation behaviour, and the makespan of the resulting schedules will be competitive with those produced by the best resource-constrained scheduling algorithms in the world.

Two aspects are however still not addressed at the end of this phase:

1. Critical paths and activity slack are absent in the schedules produced by the ACO. For the unconstrained case, the Critical Path Method (CPM) helps project planners to identify the slack of activities as well as critical paths. This allows project managers to focus their attention on the critical path as well as activities with limited slack. Unfortunately the familiar concepts of slack and critical paths are not readily available after a resource-constrained schedule optimisation has been done. With no slack or critical paths to manage, project managers will be forced to

micro-manage the project schedule, attempting to execute every activity exactly as planned. This, however, is not a desirable way to manage a project, and in most cases totally unrealistic. Bowers (1995) goes as far as saying that the lack of critical paths and activity slack in resource-constrained schedules are central to the slow adoption of these methods in practice.

2. The schedules do not account for project uncertainty. The input data of the ACO is strictly deterministic. Activities are modelled to have fixed durations, and resources capacity levels are assumed to be constant for the duration of a project. Uncertainty and disruptions are however commonplace during the engineering-planning phase, and schedules that fail to account for these factors will not produce reliable estimates. Schedules produced by the ACO therefore need to undergo some form of processing post-scheduling, that will help protect them against possible disruptions and project uncertainty.

It should therefore be clear that after the first phase of BaSE has been completed, another phase will have to be initiated to account for the above mentioned issues. The additional steps required to complete this phase will be the focus of this chapter.

Several researchers have already discussed the need for a new measure of slack in resource-constrained networks (see Willis, 1985, Ragsdale, 1989). Wiest (1964) argued that critical paths should be replaced with *critical sequences* which account for precedence links as well as resource dependencies. Bowers (1995) built on his work, and developed a generic procedure that makes it possible to identify slack and critical paths in resource-constrained schedules. His strategy involved documenting the specific resources assigned to activities during scheduling, and then inserting resource links wherever resources are transferred between activities. Artigues and Roubellat (2000) represent these resources transfers between activities with a so called *resource flow network*. Combining a resource flow network with a technological precedence network leads to a standardised way in which resource-constrained slack can be measured.

Bowers (2000) points out that various feasible resource allocations/flows exist for every schedule. Even though the resource allocation will not alter the scheduled starting times of activities, it could have a significant impact on the slack distribution and the total slack of the schedule. Both of these properties can greatly influence the outcome of a project. Distributing slack in an intelligent way, can make a schedule more robust to disruptions/uncertainty, and it could help to distribute pressure more evenly among project resources. Increasing the total slack of a schedule can greatly reduce the risk of schedule overruns, and reduce the total pressure on the project team. The more slack

a schedule has, the less chance there is of exceeding the project deadline, and the more time everyone has to complete their work.

Resource allocations therefore not only make it possible to measure resource-constrained slack in optimised schedules, but it could also be used to protect a schedule against disruptions and uncertainty. To achieve this, an appropriate slack based objective function will have to be defined for the resource allocation phase. None of the researchers mentioned above approached the problem in this manner. They rather focused their attentions on the identification of critical paths and the measuring of resource-constrained slack.

To the best of this author's knowledge, Leus and Herroelen (2002) are the only researchers that approached resource allocation as an optimisation problem. Their technique, referred to as the SBSM in this dissertation, has already been introduced to the reader in Chapter 3. They were concerned with resource allocations that lead to stable base-line schedules. Their objective was to distribute the slack in such a manner that disruptions in activity durations would have minimal affect on the starting times of other activities. Their resource allocation algorithm is based on a branch and bound procedure that incrementally adds edges to the resource flow network until a feasible flow is attained. Unfortunately their algorithm can only handle one resource type, whereas most projects in practice require multiple resource types. Even so, their work still forms an excellent basis for further research into the generation of stable base-line schedules.

This chapter will attempt to develop and incorporate a resource allocation algorithm into the BaSE framework. This will allow for resource-constrained critical paths to be identified in the ACO schedules, and also protect these schedules against possible disruptions and project uncertainty. The objective function of this resource allocation phase will be concerned with maximising the total slack available in a resource-constrained schedule. Multiple resource types will be accounted for.

This objective function is straight forward, but bodes well with the engineering-planning phase where much uncertainty stems from the availability and/or workload of resources. Engineering projects are often executed in a high pressure environment with strict deadlines. Increasing the total slack of a project will alleviate the pressure on the project team, and ultimately lower the risk of missing project deadlines. Since there has been such limited work done in this field, a heuristic strategy will be adopted as a first approach to solve the problem. Hopefully this research could serve as a basis for future work – either in the form of seed solutions for more complex optimisation procedures or as inspiration for more sophisticated heuristics.

This chapter is organised as follows: The problem statement is formulated in Section 2, along with some formal definitions and notations. Section 3 discusses the approach and methodology. A generic procedure is developed for allocating resources, and slack maximisation strategies are discussed. Eight heuristic allocation algorithms are presented which aim to maximise schedule

slack. These algorithms are then compared to each other in Section 4, and the experimental results are presented. An implementation of the algorithm is discussed in Section 5, and Section 6 will conclude the chapter.

6.2 Definitions and problem statement

In this section formal definitions and notations pertaining to the resource allocations phase will be introduced. To start, resource allocation matrices and resource induced precedence edges will be discussed. A definition of resource-constrained slack will follow, along with formulae for calculating these values. Finally the objective function and problem statement will be formulated. Note that this section builds on the notations and examples defined in the previous chapter. The reader is therefore encouraged to refer back to section 5.2 of Chapter 5 before continuing with this section.

6.2.1 Resource allocation matrices and resource edges

Traditionally, scheduling problems are not concerned with the allocation of resources. Once the scheduling phase has been completed, activity starting times are known, but it is not clear which resource should be responsible for which activity. Resource allocations are useful, since they can be used in conjunction with a project schedule to identify additional precedence constraints that are a result of resource dependencies. These resource induced precedence constraints are required to calculate the slack of a resource-constrained schedule (Bowers, 1995). It is therefore worthwhile to formally describe resource allocations[1].

The resource allocations of a resource-constrained project can be described by a $\mid R \mid \times \mid J \mid$ boolean matrix M. The row indices represent the project resources and the column numbers represent the projects activities. Each of the elements of the matrix indicate whether a resource has been assigned to an activity or not. Formally:

$$m_{rj} = \begin{cases} 0 & \text{if } r \in R \text{ is not assigned to activity } j \in J \\ 1 & \text{if } r \in R \text{ is assigned to activity } j \in J \end{cases} \tag{6.2.1}$$

A resource allocation matrix is said to be feasible if it resolves all of the resource requirements of the resource-constrained project. Formally, condition 6.2.2 needs to be satisfied:

[1]Please note that these resource allocations do not necessarily imply that a specific individual has be assigned to an activity. Proxy resources can be assigned to activities for the time being, and the actual individual that will execute an activity can be resolved at a later stage when more information is available. This concept will be discussed in detail in Chapter 7

$M_X =$

	0	1	2	3	4	5	6	7
i	0	**1**	0	0	0	0	0	0
ii	0	0	**1**	**1**	0	**1**	0	0
iii	0	0	**1**	0	**1**	**1**	**1**	0
iv	0	0	0	0	0	**1**	0	0
v	0	0	0	0	0	0	**1**	0

$M_Y =$

	0	1	2	3	4	5	6	7
i	0	**1**	0	0	0	**1**	0	0
ii	0	0	**1**	**1**	0	0	0	0
iii	0	0	0	0	**1**	**1**	**1**	0
iv	0	0	**1**	0	0	0	0	0
v	0	0	0	0	0	**1**	**1**	0

Figure 6.1: Resource allocation matrices

$$\sum_{r \in R_I} m_{rj} = n_{jI}\,, \qquad \forall j \in J \wedge \forall R_I \in Q \tag{6.2.2}$$

Let $A_r = \{j \in J \mid m_{rj} = 1\}$ denote the set of activities to which $r \in R$ has been assigned. Since a single resource can only execute one activity at a time, none of the activities contained in A_r can be executed in parallel. These activities are said to have the same *resource dependency*.

The resource allocation matrix of a project is said to be *compatible* with the project schedule if it does not prohibit the schedule from being realised if everything goes as planned. A resource allocation will be compatible with a schedule if none of the activities that need to be executed in parallel have the same resource assigned to them. If the resource allocation matrix M is compatible with the schedule S, it will be written as $M \sim S$. Formally a feasible resource allocation is compatible with a feasible schedule if condition 6.2.3 holds:

$$\sum_{j \in J_t} m_{rj} \leq 1\,, \qquad s_0 \leq t < f_{n+1} \wedge \forall r \in R \tag{6.2.3}$$

Figure 6.1 shows two different resource allocation matrices, both of which are compatible with the example project's schedule S (see figure 5.1 and 5.2).

Allocating resources to a schedule can induce additional precedence constraints. These additional precedence constraints are the result of the resource dependencies that exist between activities using the same resource(s). These resource induced precedence constraints can be represented by directed edges between activities. If a feasible resource allocation matrix has been set up that is compatible with a feasible project schedule, then these resource induced precedence edges can be derived through inspection. The resource allocation matrix can be used to identify these edges, and the schedule can be used to determine their direction. This requires further explanation: By inspecting the resource allocation matrix, edges can be inserted between all activities that require the same resource. Since $M \sim S$, no edges will exist between activities that are scheduled in parallel. To ensure that the schedule can be executed as planned, the direction of the resource induced edges need to be in accord with the project schedule. This means that every edge must be directed from the activity with the earlier starting time, to the activity with the later starting

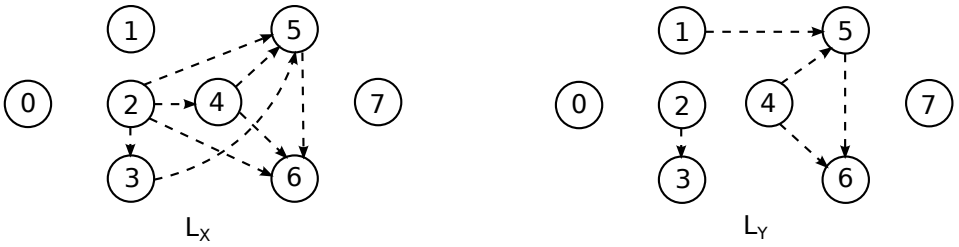


Figure 6.2: Resource induced edges

time. Formally, the resource induced precedence edges that are the result of an $M \sim S$ pairing, can be represented by a set of activity pairs:

$$L = \{(i,j) \in J \times J \mid f_i \leq s_j \wedge \exists r \in R : m_{ri} = m_{rj} = 1\} \tag{6.2.4}$$

Figure 6.2 shows the resource induced edges for $M_X \sim S$ and $M_Y \sim S$.

6.2.2 Resource-constrained slack

The CPM provides a simple method for calculating the slack values of project activities. It defines the earliest start time of an activity as the earliest point in time at which the activity can be started, should all of its predecessors be executed exactly as planned. Similarly, it defines the latest start time as the latest point in time at which the activity can be started without delaying the completion of the project.

Applying the CPM to a process model containing only technical precedence constraints will produce the unconstrained slack values of activities. The unconstrained earliest start time ES_j^μ of an activity $j \in J$ can be computed by a forward pass as follows. Starting with $ES_0^\mu = 0$ define $ES_j^\mu = max\{ES_i^\mu + d_i \mid (i,j) \in E\}$ for $j = 1, 2, \ldots, n+1$. The unconstrained latest start time LS_j^μ of an activity $j \in J$ can be computed by backward recursion from an upper bound T on the completion time of the project. Starting with $LS_{n+1}^\mu = T$ define $LS_j^\mu = min\{LS_i^\mu - d_i \mid (j,i) \in E\}$ for $j = n, \ldots, 1, 0$. The CPM defines an activity's slack as as the difference between its latest start and earliest start time, $F_j^\mu = LS_j^\mu - ES_j^\mu$.

As previously mentioned, the CPM does not consider the scheduled starting times of activities when slack values are calculated, but simply relies on the precedence constraints of the underlying project network. If the CPM is therefore applied to the schedules produced by the ACO, it will simply produce slack values corresponding to the unconstrained case. These values will however not be a true reflection of reality, due to the practical limitations imposed by resource constraints. Resource dependencies may prevent activities from starting at their unconstrained earliest starting times, and it could require activities to be finished before their unconstrained latest starting

times. The resulting slack values would therefore typically be grossly inflated. Resource dependencies need to be taken into account if realistic slack values are to be obtained. The previous section showed that it is possible to identify these resource dependencies if a feasible schedule and a compatible resource allocation matrix is available. If these resource dependencies are fed back into the process model, then the CPM can be used to calculate resource-constrained slack values. Instead of only considering technological precedence edges, the procedure will now also account for resource induced edges. Define the resource-constrained earliest start time of an activity as the earliest time instance at which an activity can be started if both technical and resource constraints are accounted for. Let the resource-constrained latest start time of an activity be defined as the latest time instance at which an activity can be started without delaying the project completion date, if both technical and resource constraints are accounted for. The resource-constrained earliest start time ES_j^S of an activity $j \in J$ can be computed by a forward pass as follows. Starting with $ES_0^S = 0$ define $ES_j^S = max\{ES_i^S + d_i \mid (i,j) \in E \cup L\}$ for $j = 1, 2, \ldots, n+1$. The resource-constrained latest start time LS_j^S of an activity $j \in J$ can be computed by a backward recursion from an upper bound T on the completion time of the project. Starting with $LS_{n+1}^S = T$ define $LS_j^S = min\{LS_i^S - d_i \mid (j,i) \in E \cup L\}$ for $j = n, \ldots, 1, 0$. Define the resource-constrained slack of an activity as $F_j^S = LS_j^S - ES_j^S$. The total slack of a resource-constrained project schedule can therefore be calculated by $SL_{tot}^S = \sum_{j \in J} F_j^S$. Let $SL_{ave}^S = \frac{SL_{tot}^S}{\sum_{j \in J} d_j}$ represent the average slack per hour of work in a resource-constrained schedule.

Take note that the computed slack will be a function of the resource allocation matrix. If resource assignments are changed in any way, then the slack will have to be recomputed. Figure 6.3 shows the computed slack for the example project. All latest start values were computed with upper bound $T = 13$. Take note how the slack values differ when different resource allocation matrices are used.

6.2.3 Problem statement

At present, two things should be clear to the reader:

1. Resource allocation is central to the calculation of slack in resource-constrained schedules.

2. The total slack and slack distribution of a schedule is influenced by the allocation of resources. (See figure 6.3)

Introducing a resource allocation phase to BaSE could therefore add immense value to the scheduling framework. It not only provides project managers with resource-constrained slack values, but it also offers the possibility to protect schedules against disruptions and uncertainty. This can be achieved

j	ES^{μ}	LS^{μ}	F^{μ}
0	0	2	2
1	0	2	2
2	0	11	11
3	0	10	10
4	2	4	2
5	8	10	2
6	8	11	3
7	11	13	2

$SL^{\mu}_{tot} = 34$

(a) Unconstrained

j	ES^{S}	LS^{S}	F^{S}
0	0	0	0
1	0	0	0
2	0	0	0
3	2	5	3
4	2	2	0
5	8	8	0
6	11	11	0
7	13	13	0

$SL^{S}_{tot} = 3$

(b) $M_X \sim S$

j	ES^{S}	LS^{S}	F^{S}
0	0	0	0
1	0	0	0
2	0	8	8
3	2	10	8
4	2	2	0
5	8	8	0
6	11	11	0
7	13	13	0

$SL^{S}_{tot} = 16$

(c) $M_Y \sim S$

Figure 6.3: Differing slack values

without any additional input besides that already required by the first phase of BaSE.

The objective function chosen for this resource allocation phase is concerned with maximising the total slack of a project. This objective function was chosen for several reasons. Firstly, it bodes well with the engineering-planning phase, where uncertainty often stems from the variable performance of resources in a pressurised environment. Secondly, it will give the reader a clear indication of the practical implication that resource allocation can have on the slack of a schedule. This is important, since limited work has been done in the field of resource allocation, with almost no mention of the impact that this process can have on project slack. This straight forward objective function will make it possible to clearly show the real life impact that resource allocation can have on the slack of a project. It also has enough practical appeal that future researchers could expand and elaborate on this work.

Although resource-constrained scheduling and resource allocation can be done in parallel, this dissertation will approach the two problems in series. The resource allocation phase will follow the resource-constrained scheduling phase, and accept the ACO schedules as input. Executing these two phases in series allows for different objective functions to be used in the separate phases, and also allows for any of the two phases to be easily replaced without affecting the other. This is important, since future researchers might develop more sophisticated algorithms for each of the respective phases.

The problem statement of the second phase of BaSE is thus as follows: Given a resource-constrained project with a feasible schedule, determine a compatible resource allocation matrix that maximises the total slack SL^{S}_{tot} of the project.

6.3 Methodology

Due to the limited work that has been done in the field of slack maximisation, a heuristic approach will be adopted as a first attempt to solve the problem. Several variations of the same heuristic will be explored and tested.

6.3.1 Generating a compatible resource allocation matrix

Before describing the proposed heuristic, it is necessary to develop a generic procedure that can be used to set up a feasible resource allocation matrix that is compatible with the project schedule. To achieve this, all of the entries of the resource requirement matrix N need to be resolved without violating condition 6.2.3. Proceed as follows: Consider each of the elements of the resource requirement matrix N. For a specific entry n_{jI}, select the required number of resources from an eligible set of resources. A resource is considered eligible if it satisfies the type requirement I, and if it has not already been assigned to activities that overlap with activity j in the schedule. Assign all of the selected resources to activity j by marking the appropriate entries in the resource allocation matrix M as 1. Resource requirements can be resolved in any order, as long as all elements of N are considered. To structure the assignment procedure, the requirements of N will be resolved in one of two ways: row by row, or column by column. In other words, each of the elements in a specific row/column of N will be resolved before moving on to the next row/column. Since the rows of N represent the project activities, the row by row ordering implies that all of the resource requirements of one activity will be resolved before moving on to the next activity. Since the columns of N represent the resource types, the column by column ordering implies that all of the resource requirements of a specific resource type will be resolved, before moving on to the next resource type. From here onwards, the row by row ordering will be referred to as the *activity-wise allocation schema*, and the column by column ordering will be referred to as the *resource-wise allocation schema*. The allocation procedure is shown for both an activity-wise allocation schema (see Algorithm 2) and a resource-wise allocation schema (see Algorithm 3).

Algorithm 2 Activity-wise allocation schema

$\xi(j, I)$: all resources of type I that have not been assigned to any activities that overlap with j.

initialise all elements of M to 0
for each activity $j \in J$ **do**
    for each resource type $I \in \{A, B, \ldots, K\}$ **do**
        for 1 to n_{jI} **do**
            calculate eligible resource set $\xi(j, I)$
            select one $r \in \xi(j, I)$
            set $m_{rj} = 1$
        end for
    end for
end for

Algorithm 3 Resource-wise allocation schema

$\xi(j, I)$: all resources of type I that have not been assigned to any activities that overlap with j.

initialise all elements of M to 0
for each resource type $I \in \{A, B, \ldots, K\}$ **do**
    for each activity $j \in J$ **do**
        for 1 to n_{jI} **do**
            calculate eligible resource set $\xi(j, I)$
            select one $r \in \xi(j, I)$
            set $m_{rj} = 1$
        end for
    end for
end for

6.3.2 Selecting a resource from the eligible set

For both of the two procedures outlined above, $\sum_{I=A}^{K} \sum_{j \in J} n_{jI}$ resource assignments will be done in total. For each of these assignments, a set of eligible resources will be computed, from which one has to be selected for assignment. If each resource assignment that needs to be done is seen as a level in a search tree, and the resources of the eligible set are viewed as branching decisions for each assignment, then a complete solution tree can be formed for a resource allocation problem. If every branch of the solution tree is explored, then the complete solution space will be covered and an optimal solution will be found. Unfortunately this is not a practical approach. Due to the combinatorial nature of the problem, solution trees would become enormous for problems of practical size. As a heuristic approach, only one of the branches of the solution tree will be explored. Intelligent branching decisions will therefore have

to be made at each level of the tree. In other words, intelligence will have to be built into the process of selecting a resource from the eligible set.

In order to understand the reasoning behind the heuristic allocation rule that was chosen, it will be necessary to explain the way in which resource allocations impact the objective function. At any point during the allocation procedure, the resource induced precedence edges can be computed based on the allocations that have already been done. Let L^i denote the resource edges that are the result of i resource assignments. At the start of the procedure, before any resource allocations have taken place, the resource induced precedence set L^0 will be empty. If the total slack of the schedule is calculated at this point, it will be the same as the total slack of the unconstrained schedule, since $E \cup L^0 = E$. Once the procedure starts doing resource assignments, resource edges will be induced, and set L^i will grow. It should be clear that after i assignments, the total slack of the schedule will either be less than, or equal to the total slack of the schedule after $i-1$ assignments. Based on this information, a resource allocation rule has been devised for selecting a resource from the eligible set ξ. The rule operates on the basis that the resource assignments that have the smallest effect on the objective function at intermediate stages, will also be good resource assignments for the global solution. This rule will be referred to as the *lowest impact resource allocation rule*. The lowest impact allocation rule can be best described with the following steps:

1. Consider the i^{th} resource assignment that needs to be done.
2. Calculate the total slack, $SL^{\varsigma*}_{tot}(E \cup L^{i-1})$, of the schedule after $i-1$ assignments.
3. Calculate the eligible resource set, ξ^i, for the i^{th} assignment.
4. Perform a mock assignment for each of the resources of ξ^i.
5. Calculate the total slack $SL^{\varsigma*}_{tot}(E \cup L^i)$ after each of these mock assignments.
6. Calculate the decrease in slack, $SL^{\varsigma*}_{tot}(E \cup L^{i-1}) - SL^{\varsigma*}_{tot}(E \cup L^i)$, that result from each of these mock assignments.
7. Select the resource whose mock assignment results in the smallest slack decrease for assignment.

Note that according to section 6.2.2, the resource-constrained slack of an activity is calculated as the difference between its resource-constrained latest start and resource constrained earliest start time. At intermediate stages of the allocation procedure, the constrained earliest start and latest start times will typically differ from the constrained earliest start and latest start times at the end of the allocation procedure. Therefore, to get a more realistic view of

the impact that a resource allocation will have on the final objective function, a slightly modified slack calculation is used for the lowest impact rule. The slack of an activity is calculated as the difference between its calculated resource-constrained latest start time and its scheduled starting time, $F_j^{\varsigma*}(E \cup L^i) = LS_j^{\varsigma}(E \cup L^i) - s_j$.

6.3.3 Determining the allocation order

When using the lowest impact rule for resource allocations, it is important to pay attention to the order in which resource allocations are done. The intermediate impact that a specific resource allocation will have on the objective function will be dependent on the resource assignments that have been done in previous steps. Different solutions will therefore be produced depending on the order in which resource allocations were done. It is therefore necessary to formally identify and control the factors that influence the order of resource allocation.

The allocation schema provides structure to the allocation process, but it does not enforce a strict ordering. It specifies whether allocations should be done activity-wise, or resource-wise, but it does not prescribe the order in which the individual activities and resource types should be processed. In other word, steps 2 and 3 of both allocation procedures still require formal structuring. From here onwards, the order in which activities are processed will be referred to as the *activity queue*, and the order in which resource types are processed will be referred to as the *resource queue*.

For this dissertation, the activity queue will be arranged according to start and end dates. Two orderings will be evaluated: Activities will either be arranged according to increasing start dates, or they will be arranged according to decreasing end dates. The former arrangement will be referred to as an *increasing activity queue*, and the latter arrangement will be referred to as a *decreasing activity queue*.

The resource types in the resource queue will be arranged according to frequency of demand. This property indicates how many activities have a non-zero demand of the resource type under consideration. Two orderings will be evaluated: Either the resource types will be arranged according to increasing frequency of demand, or they will be arranged according to decreasing frequency of demand. The former arrangement will be referred to as an *increasing resource queue*, and the latter arrangement will be referred to as a *decreasing resource queue*.

6.3.4 Heuristic resource allocation algorithms

A heuristic allocation rule has been discussed that can be used to facilitate the allocation process. The outcome of the allocation procedure will be influenced by the order in which resource requirements are resolved, and it is therefore

Table 6.1: Properties of algorithms

	Allocation schema	Activity queue	Resource queue	Allocation rule
Alg-1	Activity-wise	Increasing	Increasing	Lowest impact
Alg-2	Activity-wise	Increasing	Decreasing	Lowest impact
Alg-3	Resource-wise	Increasing	Increasing	Lowest impact
Alg-4	Resource-wise	Increasing	Decreasing	Lowest impact
Alg-5	Activity-wise	Decreasing	Increasing	Lowest impact
Alg-6	Activity-wise	Decreasing	Decreasing	Lowest impact
Alg-7	Resource-wise	Decreasing	Increasing	Lowest impact
Alg-8	Resource-wise	Decreasing	Decreasing	Lowest impact
Alg-R	Random	Random	Random	Random

necessary to do resource assignments in a meaningful order. Three structures have been identified that influence the allocation order: the allocation schema, the activity queue and the resource queue. Two unique arrangements have been proposed for each of these ordering structures. An algorithm has been implemented for each combination of these different arrangements, resulting in $2 \times 2 \times 2 = 8$ different resource allocation algorithms. A random resource allocation algorithm, Algorithm-R, has also been implemented. This algorithm allocates resources in an arbitrary order, and it selects resources for assignment at random from the eligible set. Algorithm-R essentially models the process that project managers currently follow for resource allocations, since they have no tools to assist them with this process. Although project managers would not assign resources at random, they typically do not perform resource allocations with any slack objectives in mind. With the result being similar to a random resource allocation. The properties of each of the nine algorithms are shown in table 6.1.

6.4 Experiments and Results

Four different experiments were set up to evaluate the proposed resource allocation algorithms. The first experiment offers a comparison of the performance of the eight heuristic algorithms, whereas the last three experiments aim to illustrate the practical implications of intelligent resource allocations. The setup and results of each experiment is discussed in its respective subsection.

In order to successfully execute the experiments, a sufficient number of test instances were required. As discussed in section 6.2.3, the resource allocation algorithms require the same input as the classical RCPSP, as well as a corresponding feasible project schedule. Besides these inputs, the resource allocation algorithms have no additional input parameters which have to fixed. This stands in contrast to the ACO that required extensive testing to understand the relationship between input parameters and project complexity.

The relative performance difference between the proposed resource allocation algorithms are not expected to differ much based on project complexity. Algorithm runtime should also not be a problem, since these heuristic algorithm do not suffer from the same complexity as more sophisticated optimisation algorithms. The problem instances contained in *PSPLIB* were therefore sufficient for test purposes, and no additional problems had to be generated. The *PSPLIB* is divided into four problem sets: j30, j60, j90 and j120. For the j30 problem set, projects were generated using the parameter values NC = {1.5, 1.8, 2.1}, RS ={0.2, 0.5, 0.7, 1.0}, RF = {0.25, 0.5, 0.75, 1.0} and n = 30. Ten projects were generated for every combination of NC, RS and RF, resulting in $3 \times 4 \times 4 \times 10 = 480$ problem instances. Problem sets j60 and j90 were generated in a similar fashion, the only difference being that n = 60 and n= 90 for j60 and j90 respectively. For the j120 problem set, parameter values NC = {1.5, 1.8, 2.1}, RS = {0.1, 0.2, 0.3, 0.4, 0.5}, RF = {0.25, 0.5, 0.75, 1.0} and n = 120 were used, resulting in $3 \times 5 \times 4 \times 10 = 600$ problem instances. *PSPLIB* therefore contains a total of $480 + 480 + 480 + 600 = 2040$ project instances.

All of the experiments were performed by using project instances from *PSPLIB*. The ACO discussed in the previous chapter was used to generate the corresponding project schedules. For every problem instance, the ACO analysed 1000 generations, each containing 15 ants per generation using a pheromone evaporation rate of 0.25. All other input parameters were in accord with the proposed default values discussed in section 5.5 of the previous chapter.

All algorithms and tests have been implemented in the Java programming language. A personal computer with a 3.10 GHz Intel quad core processor was used to execute the experiments.

6.4.1 Experiment 1: Comparison of heuristics

For this first experiment, tests were performed to see how the heuristic algorithms perform in relation to each other. This will provide insight into how different ordering structures influence the solution quality, and give a clear indication of which algorithm is most suited to the problem.

6.4.1.1 Setup

To gain a thorough understanding of the behaviour of the different heuristic resource allocation algorithms, every problem instance in the *PSPLIB* was analysed. Eight solutions were generated for every problem instance, one with each of the eight heuristic algorithms. The best solution to every problem instance was noted, and the deviation of every solution from the best was documented.

Table 6.2: Comparison of heuristics

	Alg-1	Alg-2	Alg-3	Alg-4	Alg-5	Alg-6	Alg-7	Alg-8
J30 (480 problems)								
Nr of best solutions found	264	253	258	249	145	142	133	131
Ave dev from best solution (%)	5.3	5.52	5.92	6.11	16.16	16.29	16.74	16.88
Ave run time per solution (s)	0.12	0.12	0.13	0.11	0.11	0.12	0.13	0.14
J60 (480 problems)								
Nr of best solutions found	141	127	153	138	111	120	50	50
Ave dev from best solution (%)	8.05	8.08	7.82	8.09	12.8	12.69	16	16.1
Ave run time per solution (s)	0.55	0.55	0.5	0.6	0.5	0.6	0.6	0.65
J90 (480 problems)								
Nr of best solutions found	100	90	142	111	121	111	26	35
Ave dev from best solution (%)	9.52	9.61	8.73	9.97	11.86	12.26	16.5	16.93
Ave run time per solution (s)	1.5	1.7	1.6	1.6	1.5	1.6	1.8	1.8
J120 (600 problems)								
Nr of best solutions found	129	124	154	168	145	132	41	45
Ave dev from best solution (%)	10.12	10.23	10.22	10.01	12.5	12.78	18.08	17.81
Ave run time per solution (s)	2.6	2.6	2.7	2.7	2.55	2.7	3	3
Overview (2040 problems)								
Nr of best solutions found	634	594	707	666	522	505	250	261
Ave dev from best solution (%)	8.36	8.47	8.29	8.63	13.28	13.46	16.91	16.99

6.4.1.2 Results

The results of Experiment 1 are shown in table 6.2. The results for each problem set are summarised, and an overview is also presented.

Remarks on Ordering Structures The results make it possible to compare the impact that different ordering structures have on the solution quality. When the influence of a specific structure is being evaluated, one should look at algorithms that only differ in regard to this one property. For example, the difference between using an increasing activity queue vs a decreasing activity queue needs to be evaluated, then the results of the algorithms inside the pairs (Alg-1, Alg-5), (Alg-2, Alg-6), (Alg-3, Alg-7) and (Alg-4, Alg-8) should be compared. When comparing these pairs, special attention should be given to the average deviation from the best solution. The number of best solutions found is relevant, but it can be misleading if an algorithm performed well for certain problems, but poorly for others. Comparisons below are therefore based on the average deviation from the best solution.

Activity Queue: An increasing activity queue proved to be superior to a decreasing activity queue for all problem sets, regardless of which allocation schema or resource queue was used. This becomes clear when a comparison is done between the results produced by the algorithms in-

side the pairs (Alg-1, Alg-5), (Alg-2, Alg-6), (Alg-3, Alg-7) and (Alg-4, Alg-8).

Resource Queue: An increasing resource queue proved to be superior to a decreasing resource queue for almost all of the problem sets, regardless of which allocation schema or activity queue was used. This becomes clear when a comparison is done between the results produced by the algorithms inside the pairs (Alg-1, Alg-2), (Alg-3, Alg-4), (Alg-5, Alg-6) and (Alg-7, Alg-8).

Allocation Schema: A resource-wise allocation schema produces superior results only when used in combination with an increasing resource queue and an increasing activity queue. For the rest of the cases, an activity-wise allocation schema produces the best results.

General Remarks

Run Time: All of the algorithms only require a few seconds to perform a resource allocation. This makes them well suited to be used as heuristics. The reader will note that there is very little difference in the run times of the respective algorithms. Since the algorithms are of equal complexity, this can be expected.

Nr of Best Solutions: The reader will note that the total number of best solutions found for every problem set exceeds the number of problem instances in the problem set. This is merely an indication that more than one of the algorithms found the best solution for some of the problem instances.

Best Algorithm: Algorithms 1 to 4 all produced comparable results, but in the end, Algorithm 3 performed the best on average. Algorithms 7 and 8 produced the worst results for all of the problem sets.

6.4.2 Experiment 2: Comparison with random resource allocation

For the second experiment, the best performing heuristic algorithm (Algorithm-3) was compared to the random resource allocation algorithm (Algorithm-R). The goal of this experiment is to illustrate that an intelligent resource allocation heuristic does indeed offer a significant increase in total slack compared to allocating resources at random.

6.4.2.1 Setup

Two tests were performed to evaluate the performance of the heuristic algorithms in comparison to the random resource allocation algorithm. In the first

Table 6.3: Comparison with Algorithm-R: Test 1

Overview (2040 problems)	Alg-1	Alg-2	Alg-3	Alg-4	Alg-5	Alg-6	Alg-7	Alg-8	Alg-R
Nr of best solutions found	634	594	707	666	522	505	250	261	27
Ave dev from best solution (%)	8.36	8.47	8.29	8.63	13.28	13.46	16.91	16.99	54.65

test, Algorithm-R was allowed to generate one solution for each of the problems in the *PSPLIB*. The results were then compared to the results of Experiment 1. If Algorithm-R consistently outperforms the heuristic algorithms, then there is no need for project managers to prefer any of the heuristic algorithms above their current procedure of resource allocations. It will be a clear indication that the heuristics are of little value.

For the second test, only the problem instances that are considered to be the most difficult to solve received attention. According to Leus and Herroelen (2002), projects with a low NC, a high RF and a high RS are generally the hardest to solve optimally[2,3]. Following this criteria, the ten most difficult problems from each of the problem sets were chosen, giving a total of 40 problems instances. Algorithm-R was allowed to generate 100 000 solutions for each of these problems. The worst solution, best solution, and average solution that Algorithm-R found for each of these problems was documented. The results were then compared to the solutions that Algorithm-3 produced for these projects. Keep in mind that Algorithm-3 only generates one solution per problem instance compared to the 100 000 that Algorithm-R is allowed to generate. Should Algorithm-3 produce superior results, then it will be a clear indication that the best heuristic is indeed of very high quality.

6.4.2.2 Results

Test 1 Table 6.3 shows the results of the first test. For the 2040 problem instances analysed, Algorithm-R only managed to find 27 of the best solutions. Algorithm-R's average deviation of 54.65% from the best solutions, gives a clear indication that the heuristic algorithms outperformed the random algorithm by a large margin.

[2]Note that resource allocation problems that are hard to solve do not correspond to the same complexity settings as problems that are hard to solve for the RCPSP. For resource allocation problems, a high RS increases the possible number of resources that can be allocated to an activity, and hence increases the complexity of the problem instance.

[3]Leus and Herroelen (2002) used the network generator *RanGen* for their experiments. Problems generated with *RanGen* have a specific order strength OS, resource factor RF and resource-constrainedness RC. This is similar to the NC, RF and RS parameters of *ProGen*. The only difference being between RC and RS: They both give an indication of how resource-constrained the project is, but they are calibrated in a different manner. A low RC value is equivalent to a high RS value. When Leus and Herroelen (2002) therefore state that a low RC value should be used to generate difficult problems, it is not at odds with what is being said here.

Table 6.4: Comparison with Algorithm-R: Test 2

	Average slack per hour of work			
	J30	J60	J90	J120
Alg-Random: Worst	0.19	0.12	0.07	0.04
Alg-Random: Best	0.44	0.29	0.25	0.08
Alg-Random: Average	0.21	0.14	0.1	0.04
Alg-3	1.11	1.08	1.28	0.25

Test 2 The results of the second test are summarised in table 6.4. For each problem instance the average slack per hour of work, SL^{ς}_{ave}, was calculated. The results were summed and averaged for each problem set. Results are presented in this format to give the reader an understanding of the practical implications that different resource allocations algorithms can have on a schedule. From table 6.4, it is clear that Algorithm-3 outperforms Algorithm-R by a large margin. For all of the problem sets analysed, Algorithm-3 generates almost three times the amount of slack as the best solutions produced by Algorithm-R. This means that even if a project manager performed 100 000 different resource allocations by hand, his best result would still produce a schedule with 3 times less slack than Algorithm-3 would produce. The significance of this result can not be overstated. Increasing the total slack of a project by a factor of 3 could have a major impact on the final outcome of a project. It could be the difference between a highly pressurised work environment, to one where resources have ample slack to compete activities. The fact that project managers typically only perform resource allocation once, and not 100 000 times will further exacerbate this situation in practice. It should therefore be clear that the total slack of a project can be significantly increased if an intelligent resource allocation algorithm is used. This could have a very beneficial outcome on the pressure of a project team, and ensure that deadlines are reached. It can therefore be concluded that Algorithm-3 is a heuristic capable of producing very good solutions.

6.4.3 Experiment 3: Comparing unconstrained and resource-constrained critical paths

For the third experiment, tests were performed to evaluate the difference between the unconstrained and resource-constrained critical path of a project schedule. At present, project managers are basing most of their strategic decisions on the unconstrained critical path, even though the resource-constrained case offers a more accurate alternative. It is important to see how these two critical paths differ in order to understand the impact that resource constraints can have on the critical path of a schedule.

6.4.3.1 Setup

In order to get a clear picture of how the unconstrained and resource-constrained critical paths differ, every problem instance in the *PSPLIB* was subject to analysis. A CPA was applied to the resource-constrained schedule before any resource allocation were done, producing the unconstrained critical path. The number of activities and working hours on the critical path were noted for each problem instance. Each of these schedules then underwent a resource allocation phase, by employing Algorithm-3. A CPA was applied to each of these schedules post resource allocation to produce the resource-constrained critical paths. Once again, the number of activities and working hours on the critical paths were noted for each problem instance. The number of overlapping activities between the unconstrained and the resource-constrained paths was also noted for each instance. This had to be done, since it is theoretically possible that the unconstrained and resource-constrained critical paths are of the same length, but consist of an entirely different set of activities.

6.4.3.2 Results

The results of experiment 3 are summarised in table 6.5 and 6.6.

Table 6.5: Comparing activities on critical paths

PS	Unconstrained (nr)	Resource-constrained (nr)	Overlap	% Increase
J30	10.54	15.47	9.44	46.77
J60	13.96	26.8	12.9	91.98
J90	16.44	37.63	15.56	128.89
J120	17.95	72.63	15.65	304.62

Table 6.6: Comparing working hours on critical paths

PS	Unconstrained (hrs)	Resource-constrained (hrs)	% Increase
J30	55.18	82.12	48.82
J60	76.39	146.63	91.95
J90	91.79	209.02	127.72
J120	100.23	402.93	302.00

These results make it very evident that resource constraints do in fact have a huge impact on the critical path of a schedule. For even the smallest problem set, both the number of activities and the number of working hours on the critical path is increased by more than 46%. These numbers increase as problem instances grow in size. For the largest problem set, an increase of more than 300% has been noted. This should be an eye opening result to all project

managers. It clearly shows that the unconstrained critical path grossly underestimates the number of critical activities in a resource-constrained project. The number of overlapping activities between the unconstrained and resource-constrained critical paths indicate that most of the activities considered critical by the unconstrained analysis are also found on the resource-constrained critical path. This result is as expected, but it is interesting to note that a small percentage of the unconstrained critical activities are in fact not critical in the resource-constrained case. This shows that the unconstrained CPA not only underestimates the number of activities on the critical path, but it also wrongly identifies a small percentage of non-critical activities as critical. The results of this experiment alone conclusively prove the necessity of resource allocations in the planning of baseline schedules.

6.4.4 Experiment 4: Practical implications of slack maximisation

The slack maximisation objective function investigated in this chapter, was specifically chosen to protect a schedule deadline against project disruptions. The idea being that a schedule with ample slack will be able to better absorb activity and resource disruptions than a schedule with very little slack. For this last experiment, this idea is put to the test by investigating the impact that project disruptions will have on a schedule with optimised resource allocations, vs a schedule with random resource allocations. This will give a good indication of whether the slack increase produced by the proposed heuristic algorithms actually offers any practical benefits for project managers.

6.4.4.1 Setup

Setting up this experiment required careful consideration. Deciding how project disruptions will be modelled, and selecting appropriate problem instances for experimentation purposes were two of the key decisions that had to be made.

Project disruptions can occur in many forms. Activities can take longer than expected, resources can become unavailable on short notice, the start date of activities can be delayed due to incomplete information etc. Most of these project disruptions can however be accurately modelled by disrupting activity durations. Take for example the case where the start date of an activity is delayed. Even though the duration of the activity is not delayed in reality, the impact on the schedule deadline can still be accurately modelled by increasing the duration of the activity by the same time increment that the activity's start date is delayed by. The same is also true for the case where a resource becomes unavailable. For this experiment, it was therefore decided to model project disruptions by means of activity duration disruptions. This also seems to be the most popular approach in academia, with several researchers making use of the disruption of activity durations to model project disruptions (see

Herroelen and Leus, 2005b). The frequency and severity of these disruptions are also points to consider. These values can differ wildly from one project to the next, depending on context and circumstances. This experiment is only interested in measuring how well the proposed heuristic slack maximisation algorithm can cope with disruptions when they do occur, and it is therefore not that crucial to model the frequency and severity of disruptions of realistic projects 100%. It was therefore decided to settle for a straight forward, but meaningful approach: disrupt the durations of a fixed number of activities by a fixed percentage. Three disruption scenarios were modelled for each problem instance, with the number of activities disrupted being increased for each scenario. The activities to be disrupted were selected at random, and the number of activities selected for each scenario was 10%, 20% and 30% (respectively) of the total number of activities of the project instance under consideration. The duration of each selected activity was disrupted by 25% of the activity's original duration. This remained fixed for all three scenarios. With this setup, it was not only possible to measure how well the heuristic could absorb schedule disruptions, but to also see how this property changes as disruptions become more frequent.

To obtain meaningful results from the experiment, it was necessary to select the appropriate set of problem instances from the *PSPLIB* for analysis. The problem instances in *PSPLIB* were initially generated for testing RCPSP algorithms. Most of these problem instances have therefore been generated to be "complex" from a resource-constrained scheduling point of view. These types of problems typically will not have much slack for activities to move around in, and the resulting schedules will therefore typically be very compact. Schedules with little to no slack should not be used to test how well a schedule can absorb disruptions, since there is very little slack available to actually absorb these disruptions to begin with. The complete set of problem instances in the *PSPLIB* were therefore not used for this experiment. Instead, the same subset of problem instances used for the second test in Experiment 2 were used for this experiment. These problem instances have a low network complexity, which reduces the number of technical precedence edges and increases the unconstrained slack, and they have a high resource strength, which implies high resource availability vs demand, which should increase the resource-constrained slack. These schedules should therefore have significantly more slack than the rest of the problem instances in *PSPLIB*, making them more suitable for a meaningful experiment.

For each of the problem instances selected for experimentation, the following steps were performed for each disruption scenario:

1. Note the resource-constrained schedule makespan.
2. Perform resource allocation.
3. Select set of activities to disrupt.

4. Disrupt activity durations.
5. Perform CPA on disrupted activity network
6. Calculate makespan of resource-constrained early start schedule.
7. Note the delay between original makespan and disrupted early start makespan.

This process was repeated twice, first using Algorithm-3 for the resource allocation phase, and then using Algorithm-R. For each problem instance, it was ensured that the set of activities disrupted remained consistent between applying Algorithm-3 and Algorithm-R. It is important to note that once a project has undergone a resource allocation phase, all resource constraints are modelled as precedence edges. It was therefore possible to modify activity durations after the allocation phase, apply a CPA, and still obtain an early/late start schedule free from resource conflicts. (This is another benefit of having a resource allocation phase in place.) As stated at the beginning of this section, all schedules for problem instances were generated using the ACO developed in chapter 5. The SSGS of the ACO will always schedule activities as early as possible. Therefore, if no activities were disrupted, and a CPA was applied to the project network, the early start schedule will correspond to the original schedule. That is why the makespan of these two schedules can be compared to compute the delay caused by disruptions. The difference in makespan delay when using Algorithm-3 versus Algorithm-R was noted for each problem instance. If the makespans of the schedules that were subject to Algorithm-3 were disrupted by the same amount as those subject to Algorithm-R, then the proposed heuristic offers no benefit to project managers. If the delays produced in the Algorithm-3 schedules are however consistently shorter than those of the Algorithm-R schedules, then the proposed heuristic does offer a very practical advantage to project managers.

6.4.4.2 Results

The results for Experiment 4 are summarised in tables 6.7, 6.8, and 6.9. The values shown are averaged out for each problem set.

Table 6.7: Absorbing disruptions: Scenario 1 (10% of activities disrupted)

		Makespan delay (hrs)		
PS	Original Makespan (hrs)	Alg-3	Alg-R	difference in delay (%)
J30	49.05	2.33	3.2	27.19
J60	66.98	3.9	5.83	33.10
J90	77.45	4.6	7.23	36.38
J120	88.9	7.01	9.8	28.47

Table 6.8: Absorbing disruptions: Scenario 2 (20% of activities disrupted)

		Makespan delay (hrs)		
PS	Original Makespan (hrs)	Alg-3	Alg-R	difference in delay (%)
J30	49.05	4.08	5.67	28.04
J60	66.98	4.98	8.03	37.98
J90	77.45	6.65	10.25	35.12
J120	88.9	11.58	15.94	27.35

Table 6.9: Absorbing disruptions: Scenario 3 (30% of activities disrupted)

		Makespan delay (hrs)		
PS	Original Makespan (hrs)	Alg-3	Alg-R	difference in delay (%)
J30	49.05	5.23	6.89	24.09
J60	66.98	8.21	10.95	25.02
J90	77.45	10.25	13.6	24.63
J120	88.9	14.33	18.83	23.90

Analysing the data in these tables make it very clear that Algorithm-3 does offer significantly better protection against schedule disruptions than randomly allocating resources. The makespan delays documented in the Algorithm-R schedules were more than 20% longer than the makespan delays documented in the Algorithm-3 schedules for all problem instances analysed. In the most extreme cases, the delay times of Algorithm-R schedules were more than 36% longer than the delay times of Algorithm-3 schedules. Project managers should welcome these results, as they show the true potential that optimised resource allocations have in the planning of baseline schedules. If a basic slack maximisation heuristic can already have such a positive influence on a schedule's ability to absorb disruptions, then there is great potential for more sophisticated allocation algorithms. Besides this result, several other interesting points can be observed. As expected, the delay times increased for both Algorithm-3 and Algorithm-R schedules as the number of disruptions were increased. The difference between the two algorithms did however not increase as the number of disruptions increased. For the first two scenarios, results were comparable, but the additional protection that Algorithm-3 provides seems to be slightly less for scenario 3. This can however be expected. As the number of disrupted activities increase, the likelihood of critical activities becoming disrupted also increases. Since there is no slack protection for critical activities, the results produced by different slack maximisation algorithms can be expected to become less noticeable as the number of disrupted critical activities increase. It's interesting to note that all of the problem sets produced comparable results. Although Algorithm-3 offers increased protection for problems sets J60 and J90 in scenarios 1 and 2, this difference was not observed for scenario 3. These results seem to suggest that the size of problem instances do not really influ-

ence the effectiveness of Algorithm-3, but more experiments will need to be done in the future to confirm this observation.

6.5 Implementation details - *ProBaSE*

The benefits of optimised resource allocation algorithms have received very little attention from academia, and it is therefore not surprising that no commercial scheduling packages implement these algorithms in any way. The experiments conducted in this chapter made it very clear that resource allocation algorithms can have a significant impact on the outcome of a project, and it is very important for project managers to be exposed to these concepts. The *ProBaSE* scheduling package introduced in the previous chapter was therefore expanded to include resource allocation functionality. Algorithm-3 was chosen for implementation, since it delivered the best results of all of the investigated algorithms. This section will highlight the resource allocation capabilities of *ProBaSE*, and will illustrate how the output can be presented to the user in a meaningful manner.

6.5.1 Required input

Similar to the ACO phase of BaSE, the resource allocation phase also requires an XML based resource-constrained project file as input. Additional scheduling info is however required before the resource allocation phase can be initiated. Start and end dates need to be provided for all activities in the project, so that a schedule that is free from resource conflicts can be formed. These scheduling dates can either be supplied as additional information in the input XML file, or it can be generated with the ACO capabilities of *ProBaSE* itself. Contrary to the ACO, Algorithm-3 has no additional input parameters that need to be adjusted, and the resource allocation phase can be initiated with the click of a button from the main window of *ProBaSE*.

6.5.2 Viewable output

The resource allocation phase provides a wealth of information that needs to be displayed to the user in a meaningful manner. *ProBaSE* divides this output into three logical units, each with its own viewing pane: Dates, Resource Usage, and Paths.

6.5.2.1 Dates

Before the resource allocation phase is initiated, some important dates will already be known. The scheduled start and end dates of every project activity will be available, and the unconstrained early start, late start and slack values

can be calculated based on the technical precedence constraints of the process model. Even though the unconstrained early/late start values are not accurate for resource-constrained projects, they still do have value. They give project managers an indication of the upper limits of activity start/end dates even if the possibility is there to outsource work or to bring in additional resources. The important resource-constrained early start, late start and slack value of each activity will however only be known once the resource allocation phase is completed. The importance of these dates have already been discussed, and it should be clear that they need to be displayed to the user. *ProBaSE* provides the user with the option to select individual activities in the Gantt chart, and will then display all of these important dates to the user as shown in figure 6.4.

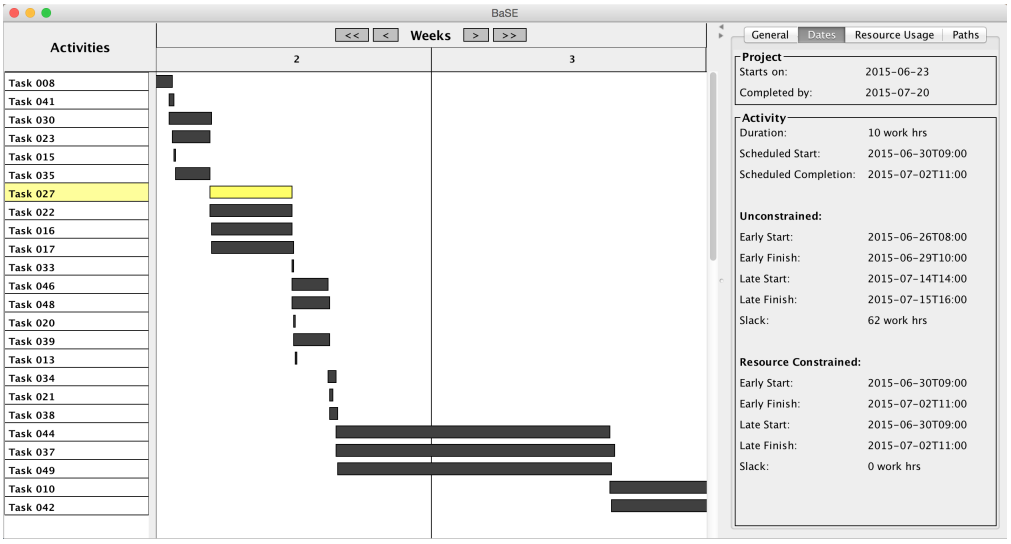


Figure 6.4: *ProBaSE* - Important Dates

6.5.2.2 Resource usage

The resource-constrained scheduling phase of the BaSE framework gives an indication of when activities need to be scheduled, but it does not give any information as to whom should be executing these activities. Assigning resources to activities is the main concern of the resource allocation phase, and *ProBaSE* displays this information to the project manager. This is very important information for staff management and long term resource planning. Figure 6.5 shows how the user can view all resource usage related information by selecting an activity from the Gantt chart.

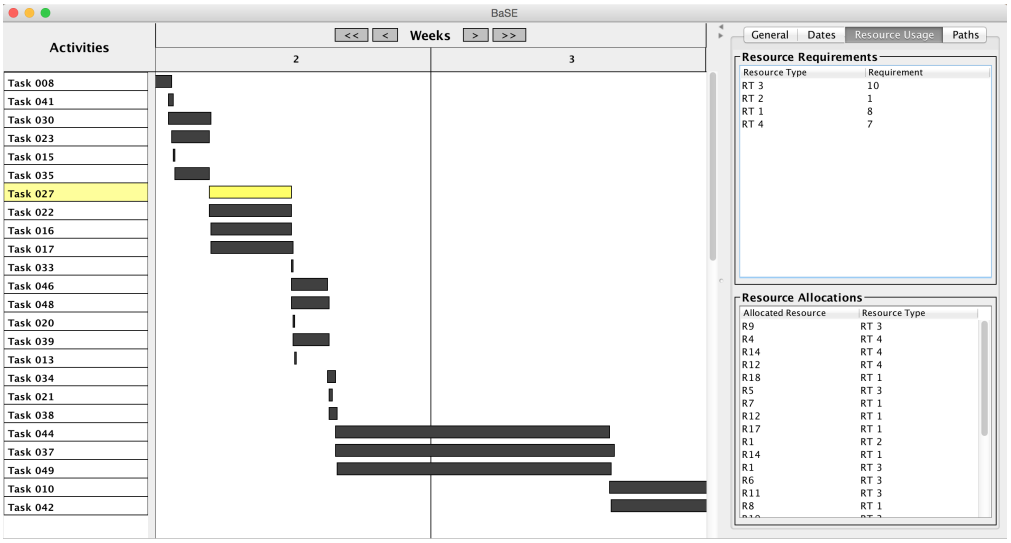


Figure 6.5: *ProBaSE* - Resource Usage

6.5.2.3 Paths

Gantt charts are excellent tools for displaying paths in project networks, and the BaSE framework produces several interesting network paths. Three main path modes are provided to the user: activity dependency paths, critical paths, and resource paths.

The activity dependency paths give an indication of the predecessors and successors of an activity. These dependencies are important because they not only indicate the logical order of activities, but they can also help project managers get a sense of how the disruption of one activity can affect other parts of the project. *ProBaSE* provides users with the choice to view technical dependencies, resource dependencies, or a combination of the two. Predecessors of an activity are indicated with green, whilst successors are indicated with orange. An example of such a dependency path is shown in figure 6.6.

The role of critical paths in project planning has been emphasized throughout this dissertation, and there is no need to further stress their importance. *ProBaSE* provides the user with the option to view either the unconstrained or the resource-constrained critical path of a project. Critical activities are displayed in red as shown in figure 6.7.

Resource paths give an indication of the course that a specific resource will follow through a project. It clearly shows the activities which the resource will be involved with, and in what order they need to be completed. This path is essentially the to-do list of a resource, and of obvious importance. These resource paths also play an important role during project execution, where they will need to be monitored to ensure the consistency of the process model. The chapter that follows will elaborate on this topic. *ProBaSE* allows users to select individual resources from a drop-down list, and displaying the

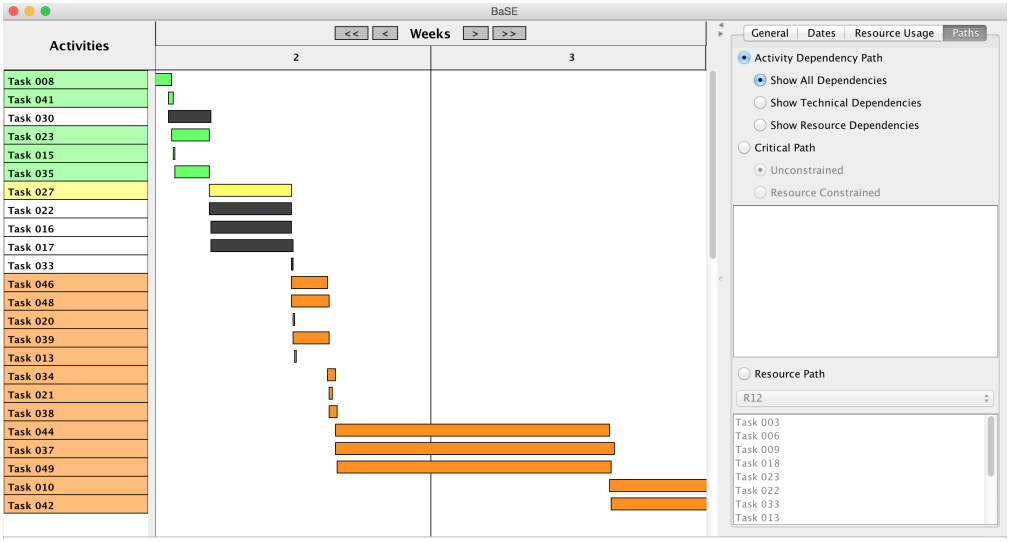


Figure 6.6: *ProBaSE* - Activity Dependency Path

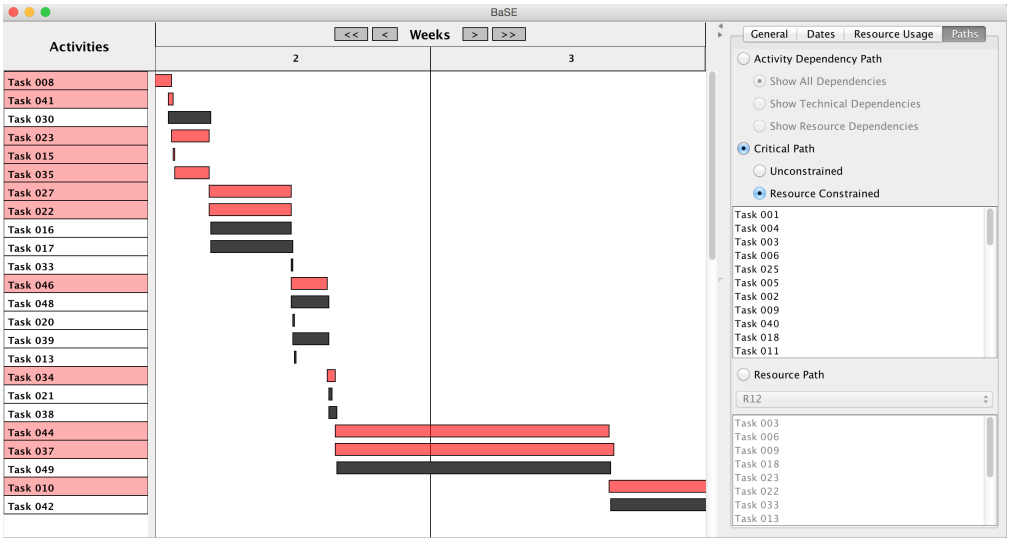


Figure 6.7: *ProBaSE* - Critical Path

corresponding activity path in blue. Figure 6.8 shows an example of such a resource path.

6.5.3 XML output

All resource allocation related information can be appended to the XML input file by saving the project from the main view in *ProBaSE*. This information can then be transferred to other commercial scheduling packages, or viewed in *ProBaSE* at a later stage. Transferring the resource allocation information to commercial scheduling packages might however not make too much sense, since most commercial scheduling packages do not treat resource dependencies

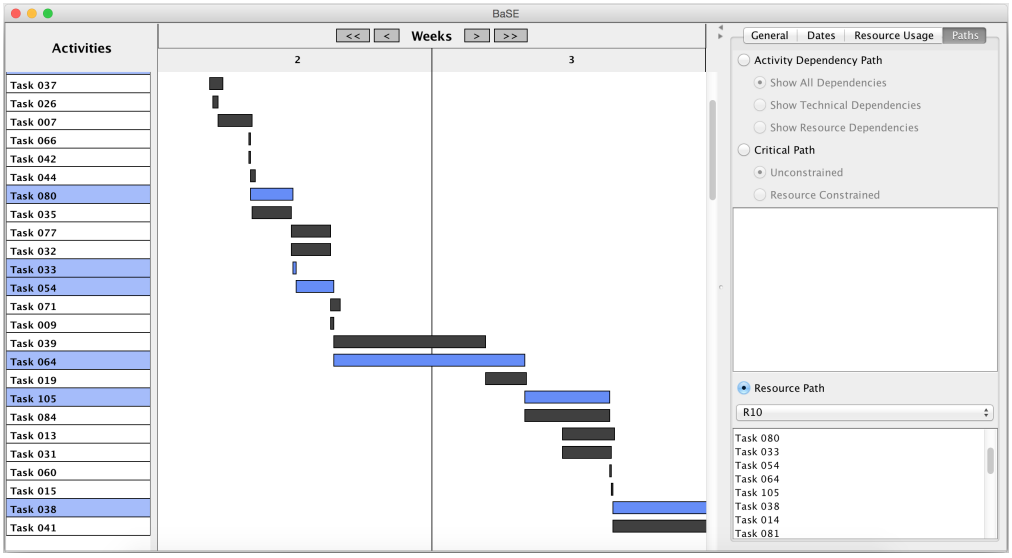


Figure 6.8: *ProBaSE* - Resource Path

as precedence edges. As a result, key information such as resource-constrained critical paths will not be readily available in these packages. Gaining access to this information was one of the main reasons for introducing the resource allocation phase to the BaSE framework and project managers will therefore have to manage this information from *ProBaSE* for the time being.

6.6 Chapter summary and recommendations

The optimised schedules produced by the first phase of BaSE do not fulfil the criteria of high quality baseline schedules. They do not offer adequate protection against project uncertainty and disruptions, and they lack resource-constrained critical paths.

The need for a standard measure of slack in resource-constrained schedules has been well documented. The work done by Bowers (1995) showed that it is indeed possible to identify slack and critical paths in resource-constrained schedules with minimal effort. The critical paths and slack that his method produced is however fixed to a specific resource allocation. Various feasible resource allocations exist for any project schedule, and they can have a significant impact on the slack distribution and total slack of the schedule. Increasing the total slack of a project will take pressure off the project team, and help protect a project against uncertainty and disruptions. The influence that resource allocation has on the slack of a schedule has received limited attention in scientific literature and it required further investigation.

This chapter therefore explored whether a resource allocation phase geared towards slack maximisation could be beneficial for the BaSE framework. Due to the limited work that has been done in this field, an heuristic approach was

proposed as a first solution to the problem. Eight heuristic resource allocation algorithms were proposed to maximise schedule slack. Each of these algorithms followed a similar strategy to allocate resources. Resources were allocated sequentially, using the lowest impact rule. The order in which resources were allocated was unique to each algorithm, and it was fixed before any allocations took place. This resource allocation order was a function of the resource queue, the activity queue, and the allocation schema. Comparing the algorithms to each other showed that the allocation order did indeed have a significant impact on the objective function. Using increasing activity and resource queues, in combination with a resource wise allocation schema delivered the best results.

Several experiments were also performed in to evaluate the practical implications of intelligent resource allocations. A random allocation algorithm was set up to mimic the resource allocation procedures used in practice. Tests were performed to compare the performance of the heuristic algorithms to this random resource allocation algorithm. All of the heuristic algorithms outperformed the random algorithm by a large margin. For the most difficult problem instances, Algorithm-3 produced almost three times more slack than the random resource allocation algorithm could produce in 100 000 attempts. This increase in slack will undoubtedly help relieve the pressure on the project team.

Comprehensive tests were also performed to evaluate the difference between unconstrained and resource constrained critical paths. The results clearly illustrate that the unconstrained critical paths that most project managers rely on are far from complete when resource constraints are introduced to the process model. By incorporating resource induced precedence edges in the process model, resource allocation algorithms provide project managers with accurate critical paths, without requiring any additional input information. The heuristic slack maximisation algorithms also provide significantly more protection against project disruptions than schedules with random resource allocations. This was demonstrated in the last experiment of this chapter.

These results should be an eye-opener for project managers. It proves that resource allocations do not only help with the identification of resource-constrained slack, but they can have a massive impact on the working environment of a project. These results were achieved by employing a basic heuristic allocation algorithm. Developing more sophisticate resource allocation algorithms can only improve the results of this study, and it will be exciting to see how these algorithms could influence the outcome of engineering projects in the future.

ProBaSE has been expanded to allow for a resource allocation phase, by implementing Algorithm-3. This provides project managers with a user friendly platform from which resource allocation can be initiated, and where key project information can be viewed. Information such as resource-constrained critical paths and resource paths which are absent from commercial scheduling packages, can easily be viewed in *ProBaSE*.

It can therefore be concluded, that with the introduction of a resource allocation phase, the BaSE framework is capable of producing schedules that conform to the criteria of high quality baseline schedules as defined in chapter 2. At present, no other scheduling framework can make this claim. The first phase of BaSE generates highly optimised schedules that are based on complete and accurate input data. These schedules are then protected against disruptions and uncertainty by maximising the total slack of the project. The resource allocation algorithm used to achieve this also has the added benefit of producing resource-constrained critical paths in these schedules. This is crucial for monitoring and control purposes, and it also helps with the identification of project risk. The development of *ProBaSE* provides the necessary implementation for this framework, and has the benefit of being open source. The real value of this system will however only become apparent once project managers start using BaSE on real projects. Their feedback would provide valuable insight into any shortcomings that the system might have, and would provide direction for the future.

Chapter 7

Maintaining the Process Model

7.1 Introduction

The BaSE framework has primarily been designed for use in the engineering-planning phase of a project. The baseline schedules produced by the method will however have an important function during project execution. The critical paths and activity slack values of a BaSE baseline schedule will be closely monitored during project execution to ensure that the project stays on track, and that the best resources are working on the most critical activities as described below.

What makes BaSE different from other scheduling frameworks, is the fact that it not only calculates activity start- and end-dates based on a specific process model, but it also alters the underlying process model during the scheduling process. The resource induced precedence edges that are produced in the resource allocation phase are fed back into the process model, and used for the calculation of resource-constrained slack and critical paths. Slack values are therefore a function of a resource allocation matrix. Altering resource allocations will therefore also alter the critical paths and slack values of a schedule. It is very important that project managers adhere to the resource allocations produced by the BaSE framework, or else the computed slack values will become meaningless.

As previously mentioned, resource allocations will typically be done with proxy resources during the planning phase, with individual project members only being assigned to activities during the execution phase. Resolving resource assignments in this manner could however induce additional resource precedence edges if certain precautions are not taken. The steps and precautions necessary to sustain the integrity of the process model during project execution will be discussed in this chapter. An extended example will be used to facilitate this process.

This chapter is organised as follows: In section 7.2, a project team will be introduced to the example project of chapter 5. Section 7.3 will discuss a

strategy for resolving resource assignments in practice, along with some basic rules to keep the process model consistent. The chapter will be concluded in section 7.4.

7.2 Introducing the project team

Refer to the example project introduced in section 5.2 of chapter 5. Recall that two resources types, R_A and R_B, were involved in the project. A resource constraint of two and three were imposed on these resource types respectively. Top level management typically derive these constraints by analysing the expected future workload of the office staff, and deciding how much capacity of each resource type can be assigned to the project under consideration. This process is normally done during the tendering phase of a project, and these constraints will be fixed once the planning phase is initiated.

To illustrate how the resource constraints of the example project could have been derived, a project team must be introduced. Table 7.1 shows such a project team along with each staff member's area of expertise. Let $Draftsman = R_A$ and $Structural\ Engineer = R_B$.

Table 7.1: Project team

Staff member	Skill
Peter	Draftsman
John	Draftsman
Mary	Draftsman
Sandy	Draftsman
Andrew	Structural Engineer
Rosy	Structural Engineer
Mark	Structural Engineer
Louis	Structural Engineer
Xola	Structural Engineer
Lumka	Structural Engineer

Four draftsmen and six structural engineers have therefore been selected to be involved with the example project. Most of these staff members will however also be involved with several other projects, and as a result, top level management has estimated that an effective capacity of only two draftsmen and three structural engineers will be available at any stage during the execution of the example project. All activities requiring a draftsman for example, would therefore be divided between the four draftsmen of the project team, but an effective output of two draftsman can be expected at any time during project execution. Assigning staff members to activities will only be done during project execution when more information is available regarding resource

availability and workload. Some form of resource allocation still however needs to be done during the planning stage to identify critical paths, and maximise the project slack. The proxy resources $R = \{i, ii, iii \ldots\}$ were therefore introduced to act as place-holders for staff members during the resource allocation phase. Let resource allocation matrix M_Y of section 6.2.1 represent the resource allocations of the example project. Table 7.2 shows the resulting to-do lists of each of the proxy resources. The to-do list for every $r \in R$ was derived by ordering the activities of each set A_r in terms of increasing start dates.

Table 7.2: Todo-list

Proxy resource	To-do list
i	1, 5
ii	2, 3
iii	4, 5, 6
iv	2
v	5, 6

Each of the staff members will be assigned a subset of a proxy resource's to-do list during project execution. This process will be referred to as *the resolving of resource assignments* from here onwards. The section that follows will introduce some basic rules for the resolving of resource assignments, to ensure the integrity of the process model.

7.3 Resolving resource assignments

Recall from the previous chapter that the resource allocation matrix M_Y leads to the formation of a set of resource induced precedence edges L_Y. The union of L_Y and the technical precedence edges E, forms the complete set of precedence edges of the process model. Resource-constrained slack values and critical paths will depend on the consistency of this edge set. The to-do list of a staff member essentially corresponds to the path that a resource will follow in the process model when executing activities. The path of a resource should be limited to existing precedence edges. If resources are transferred between activities where no precedence edges exist, then additional precedence edges will automatically be induced. This changes the structure of the process model, and could corrupt the original slack values and critical paths. Statement 1 follows from this observation:

Statement 1. *To ensure the integrity of the process model, the set of precedence edges $L \cup E$ must remain unchanged during project execution.*

This requirement can easily be satisfied by ensuring that the to-do list of a staff member is a subset of the to-do list of one and only one proxy

resource. The activities of a proxy resource's to-do list can be allocated to several staff members, but one staff member cannot execute activities from the to-do lists of several proxy resources. The mapping from staff members to proxy resources is therefore non-injective and surjective. As an example, Peter and John could therefore share the workload of proxy resource *i*, without causing any inconsistencies in the process model. This would however forbid Peter or John from being assigned to any activities of proxy resource *ii*. Table 7.3 shows an example of a valid resource assignment that would ensure the consistency of the process model during project execution.

Table 7.3: Valid resource assignment

Staff member	To-do list
Peter	1
John	5
Mary	2
Sandy	3
Andrew	4
Rosy	2
Mark	4
Louis	5, 6
Xola	5
Lumka	6

Note that staff members could also work together on activities. This is not shown in table 7.3, but Peter and John could work together on activities 1 and 5 instead of each being responsible for only one activity. This would cause no inconsistencies, as long as Peter and John still adhere to the requirement of not executing any activities of any other proxy resources besides *i*.

This requirement might however prove to be too limiting for a practical environment. The to-do lists of proxy resources are automatically generated, and project managers might find it difficult to ensure that one staff member can only work on the activities of one proxy resource. For example, if activity 2 and activity 5 are very similar activities, it might be sensible to assign them to the same staff member. This is however not possible due to the way in which activities are automatically grouped by the resource allocation algorithm. To make provision for this scenario, the requirements of statement 1 will have to be slightly relaxed. It is however possible to achieve this without sacrificing the integrity of the process model. Observe that once an activity is completed, it cannot alter the slack values of the remaining activities that still have to be completed. Completed activities and their accompanying edges can therefore be pruned from the process model without altering the remainder of the project. If these completed activities are also removed from the to-do list

of staff members, then it should be possible to adjust resource assignments to a more desired state. Take for example the situation where it would be desired to have Mary execute both activity 2 and activity 5. Since these activities belong to the to-do lists of different proxy resources, this assignment is forbidden at the start of the project. If Mary completes activity two in a timely manner, then it can be pruned from both the process model and Mary's to-do list. If resource assignments are reviewed at this point in time, it will become clear that Mary can now in fact take on activity 5 without compromising the slack values of the remainder of the project. The restriction that a staff member can only have the activities of one proxy resource on their to-do list at any point in time therefore still holds, but if these to-do lists are revised as the project progresses, then flexible resource assignment can be achieved.

7.4 Chapter summary

This chapter explored the management effort required to keep the BaSE process model consistent during project execution. The biggest concern is that activity slack values and critical paths might become corrupted during project execution. This concern stems from the fact that the slack values and critical paths produced by BaSE are specific to a resource allocation matrix. Since resource allocations are done with proxy resources during the planning phase, the danger exists that slack values might be altered if resource assignments are not resolved in a manner consistent with the proxied resource allocation matrix. This chapter proposed simple rules which can be followed to avoid this situation. These additional steps might be seen as a hindrance by some, since it requires more management effort than traditional scheduling methods. The additional effort is however well defined and requires minimal adjustment to the management procedures currently used in practice. This management effort is a small price to pay for the additional benefits offered by the BaSE framework.

Chapter 8

Conclusion and Recommendations

8.1 Conclusion

This dissertation set out to bridge the gap between scheduling in practice and scheduling in theory within the context of the civil engineering industry. The engineering-planning phase and the construction of baseline schedules received focus. Four research objectives were defined at the start of this dissertation:

1. Define the role of baseline schedules in the engineering-planning context.
2. Define the criteria for high quality baseline schedules.
3. Evaluate the state of the art.
4. Propose a scheduling framework.

Some of these objectives have previously been the subject of academic discussion, but as a unit they have never been directed at the engineering-planning phase of civil engineering projects. Not only were these four objectives met, but several new findings were also uncovered.

8.1.1 High quality baseline schedules

The first two of these objectives were addressed in Chapter 2 of this dissertation. The working environment of a typical engineering office was evaluated, and the particular needs of the planning phase were discussed. Academic literature on this topic is sparse, but most of the conclusions that were drawn regarding the working context of the engineering-planning phase should be common knowledge to most practising project managers. With the scheduling context clearly defined, it was possible to identify the specific roles of the baseline schedule within the engineering-planning phase. A baseline schedule differs from an execution schedule in the sense that it is not generated with the idea to be executed exactly as planned. It is rather a management tool that

project managers can query for critical project information. Multiple roles were therefore identified, ranging from estimation to resource planning to risk management as well as several others.

A baseline schedule needs to be versatile enough to serve all of these different purposes. Four schedule characteristics were identified as being crucial to achieving such a high quality baseline schedule. This is a very important finding, since it provides criteria against which existing and new scheduling methods can be evaluated. This not only helps direct future research, but it will also enable project managers to establish whether their scheduling software is suited for generating high quality baseline schedules or not. Such guidelines have been absent in the past, and this result should enable project managers to make more informed decisions regarding the tooling they use.

8.1.2 State of the art

A thorough investigation of the state of the art in scheduling methods was conducted in Chapter 3 of this dissertation. The criteria for high quality baseline schedules served as a basis against which the scheduling techniques used in practice as well as the scheduling techniques proposed by academia could be evaluated.

8.1.2.1 Shortcomings of commercial scheduling software

Evaluating the scheduling techniques used in practice required an analysis of the commercial scheduling software used by project managers. MS Project dominates the engineering-planning phase and became the subject of discussion in Chapter 3. Evaluating MS Project against the criteria for high quality baseline schedules revealed several of its flaws. The absence of resource-constrained slack and resource-constrained critical paths in MS Project's schedules detracts from its usefulness as a tool for generating baseline schedules. Without this information, project managers are forced to micro-manage a schedule, attempting to execute each activity exactly as planned. This process becomes unmanageable for all but the smallest projects. As a result, project managers have to rely on the unconstrained slack values and critical paths to make important management decisions. These slack values are however overly optimistic, and the critical paths incomplete. This was clearly shown in the experiments conducted in Chapter 6 of this dissertation. Project managers are therefore basing some of their most strategic decisions such as resource allocation and risk management on incomplete and erroneous data. This should be very troublesome news for most project managers.

The lack of transparency regarding the schedule optimisation algorithms used by MS Project should also be an area of major concern for all project managers. Microsoft regards the resource levelling algorithm of MS Project as proprietary, and does not disclose its inner workings. It is not even clear what

objective function the algorithm is actually attempting to solve - the smoothing of resource profiles or makespan minimisation. Under no circumstances would it be acceptable for a structural engineer to use design software that does not disclose the theory on which its calculations are based. The same should also be true for the field of project management and project scheduling. Scheduling problems are by no means trivial, and they can have a major impact on the outcome of a project. Project managers should either place pressure on Microsoft to be more transparent regarding the scheduling techniques that MS Project employs, or they should start investigating open source alternatives such as Libre Project.

8.1.2.2 Need for research

Investigating the state of the art in academic scheduling also revealed several interesting insights. The first important finding was understanding where the scheduling problems of the engineering-planning phase fit into the hierarchical-positioning framework discussed in the opening chapter of this dissertation. A critical analysis revealed that the baseline scheduling problem belonged in the tactical/operational level of the hierarchical framework, and in the high dependency, medium uncertainty region of the positioning framework. Proactive scheduling techniques are best suited for these types of scheduling problems. Research on proactive scheduling techniques is sparse, and there is much potential for future research. Chapter 3 focussed on the two most relevant proactive scheduling methods, namely the CCM and the SBSM. Even though CCM has received much more attention from popular project management literature than SBSM, it performed significantly worse when evaluated against the criteria for high quality baseline schedules. The criticism directed at the CCM in this dissertation is in line with what other researchers have found: it relies on oversimplified assumptions, and fails to provide adequate mechanisms for resolving resource conflicts. Even though the CCM provides several advantages over the traditional techniques used in practice, the assumptions that the method are based on simply do not form a solid enough foundation to build a high quality scheduling framework on. Project managers who insist on using the CCM to generate baseline schedules should at the very least be aware of the shortcomings and assumptions of the method.

The SBSM provides a refreshing alternative to the CCM. The method's innovative approach to effective slack distribution by means of intelligent resource allocation has a practical appeal which cannot be denied. The SBSM relies on deterministic input data and avoids problematic stochastic probability distributions and controversial assumptions. It requires exactly the same input as traditional scheduling methods used in practice, but produces schedules of much higher quality. Resulting schedules are not only buffered against uncertainty, but they also contain resource-constrained critical paths and slack values that are accurate. Surprisingly enough, the SBSM is one of the few

methods that realises the advantages of optimising the resource allocations of a resource-constrained schedule. Despite the obvious benefits of this process, resource allocation optimisation has been rarely discussed in academia, and has been completely ignored by practice. Researchers interested in proactive scheduling methods are encouraged to take note of the strategy employed by the SBSM, and to use intelligent resource allocations to their advantage. Unfortunately the SBSM could not be directly applied to the engineering-planning phase. The method was designed to cope with only one resource type, and the stability based objective function is not in line with the needs of the engineering-planning phase. The modular structure of the method does however lend itself out for reuse, and made it the most viable starting point for developing a scheduling framework capable of generating high quality baseline schedules for the engineering-planning phase of civil engineering projects.

8.1.3 BaSE framework

The BaSE framework introduced in this dissertation represents the first attempt towards a scheduling framework that can meet all of the unique requirements of the engineering-planning phase of civil engineering projects. The BaSE framework draws inspiration from the SBSM, and also employs a two phased system of resource-constrained scheduling followed by a resource allocation phase. The makespan minimisation and slack maximisation objective functions were specifically chosen with the working environment of the engineering-planning phase in mind. Engineering projects are typically executed in high pressure environments, with timely execution of projects being of the utmost importance. Minimising the makespan of the baseline schedule will ensure that projects are executed in the shortest possible time with the available resources. Not only will this allow companies to take on more work in a shorter time-frame, but it will assure that the largest possible time buffer is available between the planned project completion date and the project deadline. The slack maximisation objective function chosen for the resource allocation phase aims to decrease the total pressure on the project team. Resources in engineering projects typically work on multiple projects at any given moment, and will have to cope with the pressure of several deadlines of unrelated projects. Increasing activity slack will decrease the impact of activity disruptions, and lower the risk of missing the project deadline. This unique objective function has never before been discussed in the context of resource allocation, and represents a unique contribution to the field of project scheduling.

8.1.3.1 Practical enhancements to an ACO

Makespan minimisation of resource-constrained schedules have received ample attention from researchers, and it might have seemed redundant to give

the topic any more attention. The size of engineering projects are however well beyond the limits of the problem instances discussed in literature, and the topic required additional research. Meta-heuristic optimisation algorithms were deemed to be the best fit for engineering projects where the number of activities can be in excess of a few hundred. Most of these algorithms suffer from numerous input parameters, and the selection of appropriate starting values is crucial to the success of these procedures. To assist engineers with this process, it was necessary to gain a deeper understanding of the relationship between parameter values, solution quality and problem complexity.

The input parameters of a well known Ant Colony Optimisation received focus, and several new insights were uncovered. The relationship between the pheromone evaporation rate and the convergence speed of a colony was particularly revealing. It was also interesting to note that the proportional increase in solution quality is small in comparison to the proportional increase in runtime as pheromone evaporation rates decrease. Smaller projects might therefore not see so much benefit in using low pheromone evaporation rates with longer runtimes. For large projects it might however be worth waiting a day for the optimisation to complete, since it could shave a few extra weeks off the project completion date. The scale and variety of the problems sets generated for testing purposes goes beyond anything discussed in literature up to date.

The results of these experiments should provide project managers with a very good starting point for selecting input parameters for almost all of the projects that they are likely to encounter in practice. *ProBaSE* also provide project managers with a useful implementation of the algorithm that can easily be integrated with their existing management software.

8.1.3.2 The importance of resource allocation and slack maximisation

The slack maximisation objective function chosen for the resource allocation phase is unique, and required original research. The effort dedicated to formalising the resource allocation phase is a step forward, and paves the way for future research. Newly developed concepts such as the Resource Allocation Matrix (RAM) also provide project managers and researchers with a common set of terms when discussing resource allocation. The heuristic strategy adopted as a first approach for slack maximisation is sensible, and provides a valuable starting point for future researchers. The generic procedure developed for generating a feasible resource allocation matrix is not only useful for heuristic algorithms, but can also be used in the research of exact resource allocation algorithms.

Extensive testing on a large problem set revealed some valuable insights into the impact that different ordering structures can have on the outcome of the resource allocation phase. Several experiments were also performed to

investigate the practical implications of resource allocations. Comparing the heuristic allocation procedure to a random allocation algorithm made it clear that even a basic heuristic resource allocation algorithm can have a massive impact on the total slack of a project schedule. Investigating the difference between the unconstrained critical path and the resource-constrained critical path of a project also provided eye opening results. The massive discrepancies that exists between the two paths were illustrated, and for the first time, it was possible to really show just how inaccurate it would be to focus on the unconstrained critical path of a project. It was also possible to demonstrate that schedules that have been subject to slack maximisation strategies offer significant protection against project disruptions when compared with schedules where resource allocations were done at random.

The results produced by these experiments underlines the importance of resource allocation in project scheduling, and will hopefully encourage project managers to pay attention to this often overlooked phase of project planning. The development of *ProBaSE* should also help facilitate this process. Project managers now have the tools to not only perform efficient resource allocations, but they also have the possibility to analyse the resource-constrained slack and critical paths of their schedules. *ProBaSE* empowers project managers to generate baseline schedules of much higher quality than any of the commercial scheduling tools can produce today.

8.1.4 Maintaining the process model

The importance of maintaining the process model during project execution has also been brought to the attention of the reader. The resource-constrained slack values and critical paths calculated during the engineering-planning phase are always tied to a specific resource allocation matrix. The resolution of proxy resources must be in accord with this RAM, or else the original slack values calculated by BaSE are at risk of becoming corrupted. Some simple rules have been introduced in Chapter 7 that will help project managers maintain the consistency of their process model during project execution. This additional management effort might be seen as a burden by some, but it is really a small price to pay for the additional benefits of the BaSE method.

8.1.5 Industry mind shift

This dissertation succeeded in all of its research objectives. The gap between project scheduling in theory and project scheduling in practice has undoubtedly been narrowed for the civil engineering industry. The shortcomings of the scheduling techniques used in practice have been illuminated, and the advantages of BaSE have been proven. Unfortunately this study alone will most likely not be enough to change the status quo. An industry mind shift is required when it comes to project management and project scheduling. Project

managers need to accept the technical nature of scheduling problems, and approach these problems with the same mathematical rigour that civil engineers apply to their design problems. Commercial scheduling software will have no reason to introduce more sophisticated optimisation techniques if there is no industry demand for it. The onus is on academic institutions to ensure that their graduates are well trained in the fundamentals of resource-constrained scheduling and technical optimisation. It is important that the results of studies such as this dissertation are assimilated into the project management body of knowledge, and made available to the industry. The hope is that once this point is reached an increase in the success rate of civil engineering projects will follow.

8.2 Recommendations

This thesis has taken an in depth look at the generation of baseline schedules in the engineering-planning phase of civil engineering projects. Even though this thesis attempted to address the problem as thoroughly as possible, several areas are still open to further research. This section will suggest potential areas that can be explored by future researchers. To structure this section, recommendations are separated in two parts: Recommendations for resource-constrained project scheduling and recommendations pertaining to resource allocation.

8.2.1 Resource-constrained project scheduling

8.2.1.1 Testing the ACO on real life projects

Civil engineering projects often exceed the problem instances covered by academia in both size and variety. It was therefore necessary to test the ACO on a problem set beyond the limits of what has been available to date. It was important to understand how the convergence behaviour of the ACO is influenced by parameter settings and the properties of the project being solved. A problem set containing a wide variety of different projects varying in size and complexity therefore had to be generated. This was achieved by making use of the problem instance generator *ProGen*. The creators of *ProGen's* aim was to develop an instance generator that can produce projects that mimic the behaviour of real life projects. It was therefore a logical choice to make use of projects generated by *ProGen* to calibrate the ACO algorithm. Hundreds of test instance could be generated, and projects with specific properties could be tested. It is however possible that the properties of real life projects would not be as homogeneous as those instances generated by *ProGen*. Consider for example the network complexity property of a project. If *ProGen* generates a project with a network complexity of 2, then it can be expected that most areas of the

project graph will more or less display this complexity. However, if the network complexity of a real life project is measured as 2, then some areas of the project graph might display a complexity significantly higher or lower than this average. These deviations are however expected to be small, and in most cases it should not have a significant effect on the ACO convergence behaviour. It is however still necessary to test the effectiveness of the ACO on real life projects using the proposed parameter values. An attempt was made to perform such tests for this dissertation, but it proved to be difficult to obtain the input data of real life projects. Most companies regard this information as confidential and they are reluctant to give it away. Finding a sufficient number of projects for testing purposes proved to be an almost impossible task. It is therefore proposed that future researchers opt for an alternative route. One option is to modify *ProBaSE* so that it includes data capturing functionality. This functionality does not need to capture any confidential project information, but should rather gather information about the optimisation process itself. Optimisation behaviour and important project properties can be logged, and a database can be built up over time. The challenging part will however be to convince project managers to use *ProBaSE* instead of commercial scheduling packages. The financial benefit that a superior scheduling engine offers should however be reason enough for project managers to consider the usage of *ProBaSE*.

8.2.1.2 Testing the ACO performance on mega-projects

For this thesis, the ACO was calibrated for problem instances containing up to 1000 activities. For most practical cases, this will be more than sufficient. The design phase of a civil engineering project will typically have about 60 - 180 activities, and it is uncommon to hear of projects containing more than 1000 activities. There are however some so called mega-projects that could have as many as 10 000 activities. Examples include the development of new mines and underground railway networks. The performance difference between basic heuristic scheduling techniques and optimisation techniques become increasingly significant as projects grow in size. These mega-projects could therefore reap large benefits if they employ sophisticated scheduling techniques during project planning. The ACO has proved to be effective for projects of up to 1000 activities, and it is reasonable to assume that it should also perform well for such large problem instances. Several aspects however still need to be addressed by researchers. For such large problem instances the runtime of the algorithm will become significant, and it would be sensible to perform optimisations on parallel computers. At present the ants of each generation are able to run in parallel, but further enhancements might also be possible. Researchers will also have to ensure that the ACO does not run into memory issues for such large problem instances. The pheromone matrix for these mega-projects can contain as many as 100 000 000 entries, and an alternative

storage mechanism might have to be explored. There is also the question of whether researchers should try to calibrate the parameters of the ACO for such large problem instances. It could be argued that optimisation experts should be involved in the scheduling of such large projects, and that each individual project should be treated separately. This is a valid point, but even for optimisation experts it might be time consuming to experiment with different parameter values, especially for such large projects. Optimisation experts specialising in scheduling are also rare, and it might not always be possible to ensure their involvement in a project. It therefore makes sense to follow a similar route as discussed in this thesis in order to assist project managers with the selection of good parameter values for such large projects.

8.2.1.3 Accounting for multiple deadlines

In its current state, BaSE deals with time constraints in an indirect manner. The scheduling process ensures that no resource constraints are violated, whilst attempting to minimise the project duration. If this optimised schedule does not satisfy the time constraint implied by the project deadline, then project managers need to take corrective action. Two strategies are available to help project managers satisfy this constraint: Critical activities can be crashed until the project duration is sufficient, or the resource constraints will have to be relaxed and the project rescheduled. The advantage of this strategy is that it allows project managers to have complete control over which resources will be increased, and which activities will be crashed. Unfortunately this approach will not work as well for projects with multiple deadlines. Activity crashing and resource adjustments will have to be done in an intelligent manner to ensure that all deadlines are met. It will be necessary to develop more sophisticated strategies to assist project managers with this process. Before this is done however, researchers will have to determine the exact point in time at which multiple deadlines are enforced in civil engineering projects. Often times the baseline schedule will first be set up, and additional milestones and deadlines will be deduced from this schedule. If this turns out to be the case in the civil engineering industry, then BaSE can be used as is, without the addition of any further functionality.

8.2.2 Resource allocation

8.2.2.1 Investigating optimisation techniques for resource allocation

This thesis has shown that even basic heuristic algorithms can have a large impact on the total slack of a project schedule. All of the heuristic algorithms developed in this thesis outperform a random algorithm by a large margin, and their usage could already have a significant impact on the outcome of

a project. The increase in slack that these algorithms produce can reduce the pressure on the project team significantly and in effect lower the risk of missing project deadlines. It will however still be worthwhile to investigate the impact that more sophisticated optimisation procedures could have on the slack of a schedule. Researchers should develop exact solution procedures, and report on the complexity and practicality of these procedures. The work of Leus and Herroelen (2002) provides a good starting point for this research. The branch and bound procedure that they developed delivers exact solutions for projects with only one resource type, and researchers should investigate whether this algorithm can be adapted to account for multiple resource types. Undoubtedly several other techniques also exist to generate exact solutions. Researchers should investigate the solution procedures of similar problems in academia for inspiration and guidance. Unfortunately the NP-hard nature of the problem will inhibit exact solution procedures from being useful for anything but small problem instances. Most likely, projects of practical size and complexity will not be solvable by these exact solution procedures in a reasonable amount of time. For these problems it might be necessary to look at meta-heuristic procedures such as genetic algorithms, ant colony optimisations and tabu search algorithms. No work has been done in this regard, and it offers various new research opportunities.

8.2.2.2 Developing a slack centric risk model

Quantifying and measuring the risk of a project is a crucial aspect of project planning. The client needs to understand the risks involved with a project in order to justify his investment, and the engineering companies need to plan their approach based on the risk profile of a project. Several models have been developed that can be used to measure the risk of a project, but it is rare to find these models used in practice. Examples discussed in popular project management literature include PERT and the Monte Carlo simulation. These models often rely on abstract distribution functions when calculating risk, and the values they produce have no concrete meaning to project managers and clients. There is a desperate need for a standardised measure of risk in project schedules. The slack of a schedule provides a very good basis for measuring the risk of a project. Most project managers and clients are familiar with the concept of slack, and it is straight forward to measure. Two key factors can be considered:

Total slack: The risk of a project tends to increase as the total slack of a schedule decreases. Schedules with ample slack tend to absorb project disruptions much better than schedules with limited slack, and they are therefore much less likely to exceed project deadlines

Slack distribution: Slack can be used to buffer certain critical activities or resources. Ensuring that there is enough slack available for activities

with uncertain durations or activities that use scarce resources can dramatically reduce the risk of a project.

Since it is now possible to measure the slack in a resource-constrained schedule, it makes sense to develop a risk model centred around these two factors.

8.2.2.3 Exploring alternative objective functions

The slack maximisation objective function investigated in this thesis is meaningful, but there is potential for further refinement. Consider the case where the total slack of a schedule is high, but where the distribution is skewed so that only certain activities have very high slack values while others have basically no slack. This scenario is unlikely, since the structure of most project networks will typically prevent such situations from existing in the first place. As projects grow in size, it becomes much more likely that an even distribution of slack will also result in the highest total amount of slack. However, exceptions do exist, and it makes sense to have safeguards in place for these situations. To effectively deal with these situations, it will be necessary to combine slack distribution objective functions with the objective function used in this thesis. This could for example be used to buffer certain critical activities, whilst maximising the total slack of the schedule. One minor adjustment that could already improve the objective function of this thesis, is the introduction of a weight function $c(j) \in \mathbb{R}$, $j \in J$. When calculating the total slack of a schedule, the slack of certain critical/uncertain activities could be given more weight than other less important activities. The total slack objective function discussed in section 6.2.3 of Chapter 6 could therefore be rewritten as:

$$Adjusted\_SL^{s}_{tot} = \sum_{j \in J} c(j) \times F^{s}_{j} \tag{8.2.1}$$

It should be obvious that this thesis explored a special case of this objective function where all activities carries an equal weight. This reason why the weight function was never introduced, relates to the difficulty in choosing weights for activities. Should an activity that is twice as likely to exceed its duration as another activity receive twice the amount of slack? Should the activity durations play a role when weighting activities? In other words, should longer activities have more slack than shorter activities? These questions obviously require further academic attention, and a study is required that can give project managers guidance when selecting these weights. The question of risk will undoubtedly play a role in the selection of these weights. It might be more sensible to connect the objective function to the risk model discussed in the previous section. The objective function will therefore be to reduce the risk of a project by means of intelligent slack distribution. This offers several exciting new possibilities for researchers.

8.2.2.4 Developing a benchmark problem library

Resource allocation has received limited attention from academia and it is only in recent years that it has been treated as an optimisation problem. The few researchers that have done work in this field had to create their own problem instances to test their algorithms on. If this field of study does continue to grow, it will become important to have a set of standard problems that researchers can test their algorithms on. This will not only allow researchers to measure the performance of their algorithms, but it will also help project managers pick the best performing algorithms for specific problem types. Such a problem library will have to provide researchers with resource-constrained projects, as well as feasible schedules for each of these projects. Several benchmark libraries exist for the RCPSP, and it makes sense to reuse the resource-constrained projects provided by these libraries. PSPLIB, MPSPLIB and LibRCPS are examples of such libraries. It will still be necessary to generate schedules for each of these projects, but several techniques are available to do this. Ideally such a library would also provide a platform that allows researchers to submit and compare solutions. Ideally this platform would accept solutions for various objective functions. It would also be a good idea to provide researchers with lower bounds on the objective functions of each of these problems. This can help researchers identify the problems to which optimal solutions have already been found. Such a library of benchmark problems will help keep resource allocation research focused and it will motivate researchers to outperform each other.

8.2.2.5 Adapting for the construction industry

The engineering-planning phase and the construction-planning phase have many similarities. Many of the techniques developed in this thesis could therefore be reused for construction-planning. Some fundamental differences however exist and they need to be accounted for. The nature of the work that needs to be done and the resource types executing the work differ from the design phase to the construction phase. In the design phase, resources are typically personnel, and the activities they perform are mostly related to document generation and manipulation. Documents are transferred between resources, whilst the resources remain mostly stationary (personnel typically execute most of their work in the office). Document transfers are typically done via an electronic medium, and transfer times are mostly negligible. Activities can therefore be scheduled to start immediately once their predecessors have been completed, without having to accounting for transfer times of documents. This is however not the case in the construction industry. Resources are typically a combination of construction workers and machinery, and activities need to be executed in predetermined locations on site. Once an activity has been completed, resources will have to be transferred to the next location

where work needs to be completed. These transfer times cannot be ignored. The machinery that needs to be moved is often bulky, and the transfer distances can be significant. Consider the case where activity A and activity B are scheduled to be executed sequentially. Suppose that both of these activities require a crane, and that activity A needs to be executed in location X on site and activity B in location Y. If there is only one crane on site, then the crane will have to be transferred from location X to location Y. On a large construction site, the distance between location X and location Y could be significant, and it might take several hours to transfer a crane between these two locations. Ignoring these transfer times during the scheduling phase will lead to inaccurate estimations and several coordination problems.

Fortunately the model that has been developed for this thesis is already in a form that will be able to accommodate the required changes. Since resource edges are inserted whenever resources are transferred between activities, the resource transfer times can simply be modelled as edge weights. These edge weights could then be converted to activities, and inserted in the project graph once all resource allocations are done. The resulting schedule would then be a much more accurate portrayal of reality. For this procedure to work, it will however be necessary to provide additional input data to the process model. Spatial information needs to be added to activities, and the transfer times of resource between certain locations will require configuration. With the advent of Building Information Modelling (BIM), most of the spatial information could hopefully be automatically derived from CAD models, leaving only resource transfer tables to be set up.

It is important to note that the incorporation of resource transfer times will lengthen the duration of a schedule. Resources therefore need to be transferred in an intelligent manner so that this makespan increase is minimised. Minimising the project duration will therefore need to be incorporated into the objective function of the resource allocation problem. It would be sensible to combine this objective function with the slack based objective functions discussed in this paper. This will require a multi-objective approach where the goal would be to maximise the slack and minimise the duration of the schedule. Several other optimisation problems can be formulated, and researchers will need to investigate the specific needs of the construction phase to find the most applicable objective function.

8.2.2.6 Industry feedback

The model that has been developed is sound from an academic point of view. However, the concept of using resource allocation to influence the slack of a schedule is unheard of in practice. Project managers need to be exposed to these concepts, and the model and algorithms developed in this thesis need to be used on real-life projects. This will not only show project managers the benefits of applying these strategies, but it will also provide researchers with

valuable industry feedback. This feedback will illuminate the shortcomings of the model and provide direction for further research.

Bibliography

M.A. Al-Fawzan and M. Haouari. A bi-objective problem for robust resource-constrained project scheduling. *International Journal of Production Economics*, 96:175–187, 2005.

R. Alvarez-Valdes and J.M. Tamarit. Heuristic algorithms for resource-constrained project scheduling: A review and an empirical analysis. In R. Slowinski and J. Weglarz, editors, *Advances in Project Scheduling*, pages 113–134. Elsevier, Amsterdam, 1989.

C. Artigues and F. Roubellat. A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes. *European Journal Of Operational Research*, 127(2):297–316, 2000.

C. Artigues, S. Demassey, and E. Neron. *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications*. Wiley, 2008.

W. Belassi and O. Icmeli Tukel. A new framework for determining critical success/failure factors in projects. *International Journal of Project Management*, 14(3):141–151, 1996.

K. Bouleimen and H. Lecocq. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research*, 149:268–281, 2003.

G. Bounds. The last word on project management. *IIE Solutions*, 30(11): 41–43, 1998.

J.A. Bowers. Criticality in resource-constrained networks. *Journal of the Operational Research Society*, 46(1):80–91, 1995.

J.A. Bowers. Interpreting float in resource-constrained projects. *International Journal of Project Management*, 18:385–392, 2000.

C. Chatfield and T. Johnson. *Microsoft Project 2013 Step by Step*. Microsoft Press, Redmond, Washington, 2013.

D.F. Cooper. Heuristics for scheduling resource-constrained projects: An experimental investigation. *Management Science*, 22:1186–1194, 1976.

R. De Boer. *Resource-constrained multi-project management â A hierarchical decision support system.* Phd, University of Twente, Enschede, 1998.

B. De Reyck and S. van de Velde. Informatiesystemen voor projectplanning: Meer communicatie dan optimalisatie. *Business Logistics*, 99(10):104–110, 1999.

J. De Wit and W. Herroelen. An evaluation of microcomputer based software packages for project management. *European Journal of Operational Research*, 49(1):102–139, 1990.

D. Debels and M. Vanhoucke. A decomposition-based heuristic for the resource-constrained project scheduling problem. Working paper, Gent University, Gent, Belgium, 2004.

M. Deckers. Exploratief onderzoek naar het gebruik van informatiesystemen voor projectplanning. Eindverhandeling, Department of Applied Economics, K. U. Leuven, Belgium., 2001.

E. Demeulemeester and W. Herroelen. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, pages 1803–1818, 1992.

E. Demeulemeester and W. Herroelen. *Project Scheduling - A research handbook*. Kluwer Academic Publishers, Boston, 2002.

R. Dlowinski and M. Hapke. Scheduling under fuzziness, 1999.

M. Dorigo. *Optimization, learning, and natural algorithms (in Italian)*. Phd, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.

M. Dorigo and L.M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1:53–â66, 1997.

M. Dorigo and T. Stutzle. An experimental study of the simple ant colony optimization algorithm, 2001.

A. Drexel, R. Kolisch, and A. Sprecher. Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science*, 41(10):1693–1703, 1995.

A.B. Eygelaar. Resource contrained step scheduling of project tasks. Master's thesis, Department of Structural Engineering and Civil Engineering Informatics, University of Stellenbosch, South Africa., 2008.

S. Globerson. Pmbok and the critical chain. *PM Network*, 14(5):63–66, 2000.

E. M. Goldratt. *Critical Chain*. The North River Press, Great Barrington, USA, 1997.

The Standish Group. The chaos report. `http://www.standishgroup.com/sample_research_files/chaos_report_1994.pdf`, 1994.

G. Gutierez and P. Kouvelis. Parkinson's law and its implications for project management. *Management Science*, 37(8):990–1001, 1998.

S. Hartmann. A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics*, 49:433–448, 2002.

S. Hartmann and D. Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *Working Paper Series der HSBA Hamburg School of Business Administration*, (2), 2008.

W. Herroelen. Project scheduling - theory and practice. *Production and Operations Management*, 14(4):413–432, 2005.

W. Herroelen and R. Leus. On the merits and pitfalls of critical chain scheduling. *Journal of Operations Management*, 19(5):557–577, 2001.

W. Herroelen and R. Leus. Robust and reactive project scheduling: A review and classification of procedures. *Journal of Production Research*, 42(8): 1599–1620, 2004.

W. Herroelen and R. Leus. Identification and illumination of popular misconceptions about project scheduling and time buffering in a resource-constrained environment. *Journal of the Operational Research Society*, 56: 102–109, 2005a.

W. Herroelen and R. Leus. Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 165:289–306, 2005b.

W. Herroelen, R. Leus, and E. Demeulemeester. Critical chain project scheduling: Do not oversimplify. *Project Management Journal*, 33(4):48–60, 2002.

A. Kastor and K. Sirakoulis. The effectiveness of resource levelling tools for resource constraint project scheduling problem. *International Journal of Project Management*, 27:493–500, 2009.

M. Keil, A. Rai, J. E. C. Mann, and G. P. Zhang. Why software projects escalate: The importance of project management constructs. *IEEE Transactions on Engineering Management*, 50(3):251–261, 2003.

H. Kerzner. *Project management. A systems approach to planning, scheduling and controlling*. Wiley, 1998.

P. Kobylanski and D. Kuchta. A note on the paper by m. a. al-fawzan and m. haouari about a bi-objective problem for robust resource-constrained project scheduling. *International Journal of Production Economics*, 107: 496–501, 2007.

R. Kolisch. Resource allocation capabilities of commercial project management software packages. *Interfaces*, 29(4):19–31, 1999.

R. Kolisch and S. Hartmann. *Handbook on Recent Advances in Project Scheduling*, chapter Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis, pages 197–212. Kluwer, Amsterdam, The Netherlands, 1999.

L.P. Leach. *Critical Chain Project Management.* Artech House Publishers, 2000.

R. Leus. *The generation of stable project plans.* Phd, Katholieke Universiteit Leuven, 2003.

R. Leus and W. Herroelen. Stability and resource allocation in project planning. Technical Report 0256, Operations Management Group, Department of Applied Economics, Katholieke Universiteit Leuven, 2002.

M.J. Liberatore and B. Pollack-Johnson. Factors influencing the usage and selection of project management software. *IEEE Transactions on Engineering Management*, 50(2):164–174, 2003.

M.J. Liberatore, B. Pollack-Johnson, and C.A. Smith. Project management in construction: software use and research direction. *Journal of Construction Engineering Management*, 127:101–107, 2001.

J.R. Meredith and S.J Jr. Mantel. *Project management. A managerial approach.* Wiley and Sons, 2000.

D. Merkle, M. Middendorf, and H. Schmeck. Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation*, 6:333–346, 2002.

R.C. Newbold. *Project management in the fast lane - Applying the Theory of Constraints.* The St. Lucie Press, Boca Raton., 1998.

K. Nonobe and T. Ibaraki. Formulation and tabu search algorithm for the resource constrained project scheduling problem (rcpsp). Technical report, Department of Applied Mathematics and Physics, Kyoto University, Japan, 1999.

C.N. Parkinson. *Parkinson's Law.* The Riverside Press, Cambridge, 1957.

T.L. Pascoe. Allocation of resources c.p.m. *Revue Francaise Recherche Operationelle*, 38:31–38, 1966.

F.S. Patrick. Critical chain scheduling and buffer management - getting out from between parkinson's rock and murphy's hard place. *PM Network*, 13 (April):57–62, 1999.

J. K. Pinto and S. Mantel. The causes of project failure. *IEEE Transactions on Engineering Management*, EM-37:269–276, 1990.

J. K. Pinto and J.E. Prescott. Planning and tactical factors in the project implementation process. *Journal of Management Studies*, 27(3):305–327, 1990.

J.K. Pinto. Some constraints on the theory of constraints - taking a critical look at the critical chain. *PM Network*, 13(August):49–51, 1999.

PMI. *A Guide to the Project Management Body of Knowledge (PMBOK Â® Guide)*. Project Management Institute, Newton Square, Pennsylvania, 5 edition, 2013.

C. Ragsdale. The current state of network simulation in project management theory and practice. *OMEGA International Journal of Management Science*, 17(1):21–25, 1989.

G.K. Rand. Critical chain: the theory of constraints applied to project management. *International Journal of Project Management*, 18(3):173–177, 2000.

R.J. Schonberger. Why projects are always late: a rationale based on manual simulation of a pert/cpm network. *Interfaces*, 29(5):66–70, 1981.

J. Schuyler. Tip of the week 26: Critical chain. `http://www.maxvalue.com/tip026.htm.`, 1997.

J. Schuyler. Tip of the week 39: Project management in the fast lane: Applying the theory of constraints. `http://www.maxvalue.com/tip039.htm.`, 1998.

S. Van de Vonder, E. Demeulemeester, W. Herroelen, and R. Leus. The use of buffers in project management: The trade-off between stability and makespan. *International Journal of Production Economics*, 97(2):227–240, 2005.

J.D. Wiest. Some properties of schedules for large projects with limited resources. *Operations Research*, 12(May-June):395–418, 1964.

R.J. Willis. Critical path analysis and resource-constrained scheduling - theory and practice. *European Journal of Operational Research*, 21:149–155, 1985.

G. Winch. Thirty years of project management what have we learned? In *British Academy of Management Conference Proceedings*, pages 8.127–8.145, Birmingham, UK, 1996. Aston Business School.

B.M. Woodworth and S. Shanahan. Identifying the critical sequence in a resource constrained project. *International Journal of Project Management.*, 6(2):89–96, 1988.

E. Yourdon. *Death March*. Yourdon Press, 2 edition, 2003.