FACULTEIT ECONOMISCHE EN
TOEGEPASTE ECONOMISCHE
WETENSCHAPPEN

KATHOLIEKE
UNIVERSITEIT
LEUVEN

Proactive-reactive procedures for robust project scheduling

Proefschrift voorgedragen tot
het behalen van de graad van
Doctor in de Toegepaste
Economische Wetenschappen

door

Stijn VAN DE VONDER

Number 247 2006

Doctoral committee

Advisor:	Prof. dr. Willy Herroelen	Katholieke Universiteit Leuven
Members:	Prof. dr. Jean-Charles Billaut	Université François-Rabelais Tours
	Prof. dr. Erik Demeulemeester	Katholieke Universiteit Leuven
	Prof. dr. Marc Lambrecht	Katholieke Universiteit Leuven
	Prof. dr. Roel Leus	Katholieke Universiteit Leuven
	Prof. dr. Mario Vanhoucke	Universiteit Gent

Uncertainty is the only certainty,
and knowing how to live with
insecurity is the only security.

John Allen Paulos

Dankwoord

Waarschijnlijk ben ik niet de persoon die het veelvuldigst zijn dankbaarheid uit en durf ik de voorspoed die ik al mijn hele leven beleef wel eens als vanzelfsprekend beschouwen. Dit neemt niet weg dat ik enorm dankbaar ben ten opzichte van een aantal personen. Dit dankwoord is een uitgelezen gelegenheid om deze mensen eens expliciet in de bloemetjes te zetten voor alles wat ze voor mij betekend hebben gedurende het tot stand komen van dit doctoraal proefschrift.

Vooreerst komt mijn promotor *Prof. Dr. Willy Herroelen*, de godfather van de Belgische projectplanningsfamilie. Vier jaar lang heb ik de eer gehad om met al mijn vragen bij deze autoriteit in ons vakgebied langs te kunnen gaan. Slechts wanneer tijdens één van de talrijke internationale conferenties waaraan we beroepshalve deelnamen[1] , een schuchter buigende Aziaat zijn idool mistel Helloëlen komt groeten, besef je wat een privilege dat is. Maar voor ons is Prof. Dr. Willy Herroelen gewoon Willy. Het levende bewijs dat de veelbesproken generatiekloof een fabel is. Drie wetenschappelijke zonen van de godfather zijn mij voorgegaan. Alle drie hebben ze nu een mooie academische carrière. Ik hoop dat binnen ettelijke jaren Willy ook met fierheid zal terugkijken op wat de jongste telg gepresteerd heeft.

Ten tweede is er *Prof. Dr. Erik Demeulemeester*. Klaar om kortelings de fakkel als "Don Scheduling" over te nemen. Officieel is Erik een lid van mijn doctoraatscommissie, maar in praktijk is zijn bijdrage veel groter dan dat. Erik adopteerde me alsof ik zijn eigen assistent was. Steeds kon ik terugvallen op zijn enorme inzicht in project scheduling en programmatuur.

[1] Dit tafereel speelde zich meerbepaald af op een met tequila overgoten receptie onder de palmbomen op de parelwitte stranden van Cancún, Mexico.

Een doctoraat schrijven mag dan een hele opgave zijn, maar met Willy en Erik aan je zijde ben je al half gewonnen voor je bent begonnen.

Professor Dr. Marc Lambrecht heb ik leren kennen als een uiterst interessante man. Ik acht het persoonlijk hoog dat hij nooit de praktische relevantie van onderzoek uit het oog verliest. Van bij het prille begin heeft hij mijn doctoraat mee begeleid en ondanks zijn drukke agenda vond hij steeds gaatjes om een babbeltje te maken. Want over één ding is iedereen het eens: Marc is een geboren entertainer.

Professor Dr. Roel Leus heb ik zien evolueren: van assistent, via dr., tot professor. Steeds is hij een zeer toegewijde onderzoeker geweest. Oog voor detail is hem niet vreemd. Streng, maar rechtvaardig. Indien twee gelijkdenkenden voor één tellen, tellen Roel en ik voor twee. Daarin zit nu net de enorme meerwaarde die hij gehad heeft bij het nalezen van mijn papers en van dit proefschrift.

Je tiens également à remercier le *Professeur Dr. Jean-Charles Billaut* de l'université François-Rabelais de Tours d'avoir accepté de faire partie de la commission du jury. C'est un honneur pour moi d'accueillir une personne d'une telle renommée. Les longues heures de trajet ferroviaire n'ont pas eu raison de sa volonté d'être présent aux défenses préliminaire et publique et je l'en remercie.

Professor Dr. Mario Vanhoucke van de Universiteit Gent leerde ik tijdens de eerste conferentie van mijn academische carrière kennen tussen pot en pint. Pas later leerde ik hem kennen als begenadigde wetenschapper en uiteindelijk als extern lid van mijn doctoraatscommissie. Ik heb steeds enorm respect gehad voor de onderzoeksgroep die hij in Gent opgebouwd heeft. Hij en zijn assistenten *Dieter*, *Broos* en *Vincent* zijn geen werknemers van een concurrerende universiteit, maar onze vrienden uit Gent die samen met onze vakgroep ons klein landje verdedigden op internationale conferenties... vaak tot in de vroege uurtjes.

Elke dag naar het werk gaan hoeft geen marteling te zijn. Zeker niet als uw collegae na verloop van tijd ook uw vrienden worden. Stuk voor stuk capabele mensen met een mooie persoonlijkheid. In de twee jaar dat *Stefan* bij ons werkt, heeft hij een centrale plaats ingenomen in onze onderzoeksgroep. Het is wetenschappelijk bewezen: *everybody loves Stefan.*

Als bureaugenoot leerde ik hem ook kennen als een behulpzaam iemand en een onderzoeker "pur sang", die desnoods eten en slaap laat vallen voor zijn onderzoek. Ook met *Olivier* deelde ik gedurende één jaar een bureau. In mijn ogen incarneert Olivier het prototype van de succesvolle zakenman en ik gun hem dit van harte. Aan alle andere collega's houd ik eveneens uitsluitend goede herinneringen over. *Brecht*, *Damien*, *Filip*, (Geo-) *Jeroen*, *Kristof*, *Robert*, nieuwkomers *Eline* en *Lu* en natuurlijk ook mijn basketball buddy *Jade*. Bedankt aan allen voor al deze mooie tijden.

Gedurende het laatste jaar van mijn doctoraat heb ik de kans gehad om deel uit te maken van het onderzoeksproject "Risicomanagement in de bouw". Een uitgelezen opportuniteit om onze theoretische verwezenlijkingen aan de realiteit te toetsen. Voor een praktische denker als ikzelf was dit project een welgekomen verfrissing. De samenwerking met het wetenschappelijk technisch centrum voor bouwbedrijf (WTCB) verloopt steeds beter, wat me doet hopen dat de kloof tussen theorie en praktijk geen onoverbrugbare vallei is. Met deze wil ik de medewerkers van het WTCB en de talrijke bedrijven die hun geloof in onze methodologie hebben geuit, bedanken. Speciale dank hierbij gaat uit naar *Anton Boone* (WTCB), die de drijvende kracht achter het project is. Tevens wil ik ook mijn collega *Damien Schatteman* bedanken voor het project van binnen onze onderzoeksgroep in goede banen te leiden. Damien is als geen ander gemotiveerd om het hele project te doen slagen. Het negende hoofdstuk in deze thesis is in samenwerking met hem tot stand gekomen.

I am also grateful to *Professor Dr. Francisco Ballestín* (University of Navarra, Spain) for the joined work while he was visiting our university. He greatly assisted me in the work performed in chapters 6 and 7. After his stay in Leuven, we experienced the difficulties of co-working over the Internet. I wish him all the best for his academic career.

Mijn ouders mogen natuurlijk niet op de lijst ontbreken. Ik merk dat ze deze dagen enorm fier op me zijn. Maar eigenlijk zouden ze fier op zichzelf moeten zijn, want het feit dat ik mijn doctoraat nu kan verdedigen is het resultaat van de opvoeding en de kansen die ze mij gegeven hebben. Slechts een week voor mijn 27ste verjaardag heb ik het ouderlijke nest verlaten. Gedurende al die jaren onder hun vleugels hebben mijn ouders me altijd de

nodige vrijheid gegeven zonder daardoor interesse in mij te verliezen. Ook mijn zus *Katrien* kan daarover meespreken. Sinds de dag van mijn geboorte word ik door haar beschermd en bemoederd en dat gebeurt nog steeds. Nog veel minder dan iemand anders bedank ik haar voor wat ze allemaal voor haar broertje doet. Laat me toe om dit bij deze toch eens te doen.

Ten laatste wil ik nog een hele resem personen bedanken die niet mogen ontbreken in dit dankwoord: mijn beste vriendin, mijn flatgenote, mijn privé-kok, mijn chauffeur, mijn fitnesspartner, mijn lerares Frans, mijn winkelexperte, mijn vaste reisgezel naar het pittoreske La Roche-en-Ardenne, mijn knuffel, mijn steun en toeverlaat, mijn grote liefde,. . . , mijn *Geneviève*. Alles in één, de belangrijkste persoon in mijn leven. Zonder jou zou ik niet zijn wie ik ben. Zonder jou zou dit proefschrift niet zijn wat het is.

Stijn Van de Vonder

Leuven, 22 December 2006

Abstract

The vast majority of research efforts in project scheduling concentrates on developing procedures to generate workable baseline schedules that minimize the project makespan in a deterministic environment. However, a real-life project is inevitably subject to uncertainty during execution. This dissertation aims to introduce procedures that cope with disruptions during execution. We limit ourselves to the treatment of *time uncertainties* caused by the fact that actually realized activity durations during project execution may deviate from the expected activity durations.

When dealing with uncertainty in a scheduling environment, there are, generally spoken, two main approaches. *Proactive scheduling* focuses at incorporating safety in the schedule to absorb future disruptions, while *reactive scheduling* denotes how to react when a disruption occurs. Both approaches are inescapably related. We will investigate how several proactive-reactive scheduling decisions can help a project manager to increase the quality of a project.

The text of this dissertation is organized as follows. *Chapter 1* introduces the problem of proactive-reactive project scheduling and situates it in the extensive project scheduling literature. *Chapter 2* formulates the problem at hand and defines the concepts required in the remainder of the thesis. The trade-off between makespan and stability in project scheduling of *Chapter 3* justifies the research efforts made in Chapters 4, 5 and 6 to add safety to the baseline schedule. Mainly two approaches to add safety are discussed in this thesis. First, a schedule is made proactive in *Chapter 4* by deciding how the resources flow throughout the project. Next in *Chapter 5*, we develop efficient and effective procedures to add idle time (buffers)

into a schedule to anticipate unforeseen events. *Chapter 6* contains an extension of Chapters 4 and 5 by merging scheduling, resource allocation and buffer allocation decisions into an integrated approach. The reactive procedures that are required to decide how to react to disruptions that cannot be absorbed by the baseline schedule are introduced in *Chapter 7*. In *Chapter 8* an extensive simulation-based experiment is set-up to evaluate several predictive-reactive resource-constrained project scheduling procedures proposed in the previous chapters. *Chapter 9* applies the proactive-reactive project scheduling methodology to a real-life project that stems from our experience in the Belgian construction industry. Accordingly, a risk management framework is introduced to detect and analyze the risks that constitute the uncertainty implied in the project. In a last chapter, some overall conclusions and recommendations are provided.

Samenvatting

Het merendeel van de onderzoeksinspanningen in projectplanning heeft de ontwikkeling van een werkbaar basisplan met een zo kort mogelijke duurtijd van het project tot doel. Hierbij wordt vaak uitgegaan van de veronderstelling dat de duurtijden van de activiteiten gekend en deterministisch zijn. Een realistisch project zal echter steeds onderhevig zijn aan verstoringen. In deze thesis worden procedures voorgesteld die helpen om met zulke verstoringen om te gaan. We beperken ons onderzoek tot de behandeling van onzekerheid ten gevolge van activiteiten die tijdens de uitvoering langer of minder lang blijken te duren dan oorspronkelijk verwacht.

Men kan twee denkrichtingen onderscheiden om met onzekerheid om te gaan in een project. Proactief plannen probeert op toekomstige verstoringen te anticiperen door veiligheid in het projectplan in te bouwen, terwijl de reactieve tegenhanger voorschrijft hoe te reageren op een zich voordoende verstoring. Beide benaderingen zijn onvermijdelijk verbonden. In deze thesis wordt onderzocht hoe proactieve en reactieve planningsbeslissingen projectmanagers kunnen helpen om de kwaliteit van een project te verhogen.

De indeling van de thesistekst is als volgt. *Het eerste hoofdstuk* leidt het proactieve-reactieve planningsprobleem in en geeft een situering in de reeds bestaande literatuur omtrent projectplanning. *Hoofdstuk 2* geeft een formele beschrijving van het onderzochte probleem en definieert een aantal concepten die in de volgende hoofdstukken veelvuldig zullen gebruikt worden. De afweging tussen de verwachte duurtijd en de robuustheid als prestatiemaatstaven van een projectplan wordt besproken in *Hoofdstuk 3* en zet ons aan tot de verdere ontwikkeling van robuuste planningsmethoden in Hoofdstukken 4, 5 en 6. Twee benaderingen om veiligheid in een basis-

plan in te bouwen worden besproken in deze thesis. Ten eerste wordt een plan in *Hoofdstuk 4* proactief gemaakt door vast te leggen hoe de vereiste hulpmiddelen tussen de verschillende activiteiten van het project circuleren. Vervolgens worden in *Hoofdstuk 5* efficiënte en effectieve algoritmes ontworpen om buffers in een plan in te voegen om te anticiperen op risico's. *Hoofdstuk 6* bespreekt een uitbreiding van Hoofdstukken 4 en 5 waarin het eigenlijke plannen, de hulpmiddelentoewijzing en het toevoegen van buffers geïntegreerd worden. De reactieve procedures die zorgen voor een toepasselijke reactie op verstoringen die niet kunnen opgevangen worden door de ingebouwde veiligheid worden in *Hoofdstuk 7* aan een grondige studie onderworpen. In *Hoofdstuk 8* wordt door middel van simulatie een uitgebreid experiment opgezet om verscheidene proactieve en reactieve technieken uit de vorige hoofdstukken te analyseren en evalueren. *Hoofdstuk 9* past de methodologie van het proactief-reactief plannen toe op een reëel project uit de Belgische bouwnijverheid. Hiervoor wordt er vooreerst een structurele methode voor risicodetectie en risicoanalyse vooropgesteld, die de projectmanager helpt om de inherente onzekerheid te kwantificeren. Het proefschrift wordt besloten met een aantal algemene conclusies en aanbevelingen.

Contents

Chapter 1

Introduction

The research field of proactive-reactive project scheduling is just emerging. Before giving a rigorous definition of the proactive-reactive project scheduling problem (Chapter 2), this chapter positions the research efforts made in this dissertation into the broad project scheduling literature. A first section is devoted to the concepts of *project management* and *project scheduling*. Afterwards, the standard problem in project scheduling is shortly revisited. We conclude this chapter with a short introduction of *stable project scheduling*, the sub-domain of project management that is the subject of this dissertation.

1.1 Project management and scheduling

Projects are of all times. The ancient Egyptian pyramids and the Maya temples are often considered as the world's first large projects. Nowadays, a report of the Project Management Institute (PMI 2003 at www.pmi.org) estimates that businesses spend $2.3 trillion annually on projects in the U.S. alone. The growing interest in the field of project management has resulted in many new theories, techniques and computer applications to support project managers in achieving their objectives. However, a survey by the Standish Group International (www.standishgroup.com) in 2003 showed that only 34% of all examined recent projects finished within time and budget, which is already a substantial improvement compared to the

16% success rate measured in the 1994 report. The average cost overrun decreased from 180% in 1994 of the project budget to 43% in 2003 thanks to a greater awareness of the relevance of project management. However, most projects are obviously still far from being perfectly managed.

Project scheduling is the part of project management that involves the construction of a *baseline schedule* which specifies for each activity the precedence and resource feasible start times that will be used as a baseline for project execution. Such a schedule helps to visualize the project and is a starting point for both internal and external planning and communication. Careful project scheduling has been shown to be an important factor to improve the success rate of the project. An internal study by Maes et al. (2000) has found that inferior planning is the third[1] reason of company failure in the Belgian construction industry. This incites researchers to further develop new project scheduling methods.

1.2 The RCPSP

Most research done in resource-constrained project scheduling concentrates on minimizing the project duration in either deterministic or stochastic environments. *The resource-constrained project scheduling problem* (RCPSP) aims to minimize the duration of a project subject to finish-start zero-lag precedence constraints (see Section 2.1.1) and renewable resource constraints (see Section 2.1.3) in a deterministic environment. Blazewicz et al. (1983) have shown that the RCPSP is NP-hard in the strong sense. Many exact and heuristic algorithms have been described in the literature to construct workable *baseline schedules* that solve the deterministic RCPSP. For extensive overviews we refer to Herroelen et al. (1998), Kolisch & Padman (1999), Kolisch & Hartmann (1999), Brucker et al. (1999) and Demeulemeester & Herroelen (2002).

[1] It is only preceded by (1) inadequate human capital and (2) mismanagement.

1.3 Uncertainty in project scheduling

During project execution, however, a real-life project will never execute exactly as it was planned due to uncertainty. Many definitions of uncertainty have been proposed since Knight (1921) introduced this concept on the economic scale. Uncertainty can originate form multiple sources. It can be ambiguity resulting from subjective estimates that are prone to human errors or it can be variability arising from unexpected events or *risks*. Risks are defined by Knight as follows:

A risk is present when future events occur with measurable probability.

Remark that the common understanding tends to presume that these risks induce a negative impact on the project when occurring, otherwise the term opportunity is often used. More recent definitions of risk do however specify a positive or negative effect on project objectives (cf. PMI at www.pmi.org).

Risks can take many forms. Resources can become temporarily unavailable (see Lambrechts et al. (2006ab) and Drezet (2005)), activities may have to be inserted or dropped (Artigues & Roubellat 2000), due dates may change, activities may take longer or less long than original expected, etc. The common practice of dealing with these uncertainties by taking deterministic averages of the estimated parameters might lead to serious fallacies (Elmaghraby 2005).

The *stochastic RCPSP* (Stork 2001) is an extension of the RCPSP with stochastic activity durations. Because of the stochastic nature of this problem, a *schedule*, which is a list of activity starting times, does no longer contain all the required information for a solution to this problem. A solution for the stochastic RCPSP needs to define the appropriate reactive action for every possible disruptive event during project execution, given the current state of the project and the a priori knowledge of future activity distributions. Möhring et al. (1984, 1985) call such a reactive way of generating a solution a *scheduling policy* or *scheduling strategy* and give a complete characterization of policies and corresponding subclasses. In pure *dynamic scheduling* (Stork 2001), the use of schedules is even eliminated altogether. A policy makes dynamic scheduling decisions during project execution at

stochastic decision points, usually corresponding to the completion times of activities. Stork implemented branch-and-bound procedures to minimize the expected makespan for different classes of policies. Research on heuristic procedures for solving the stochastic RCPSP is also just emerging (Golenko-Ginzburg & Gonik (1998), Ballestin (2006),...).

However, the absence of a schedule has some consequences from an economic point of view. The *baseline schedule* (pre-schedule, predictive schedule) namely serves a number of important functions (Aytug et al. (2005), Mehta & Uzsoy (1998)), such as facilitating resource allocation, providing a basis for planning external activities (i.e. contracts with subcontractors) and visualizing future work for employees. By consequence, a baseline schedule often needs to be sought before the beginning of the project as a prediction of how the project is expected to unfold.

Yang (1996) claims that using a baseline schedule accompanied with a dispatching rule, i.e. proactive-reactive scheduling, often leads to a lower expected makespan than pure dynamic scheduling procedures. Recently, Ballestín (2006) also remarked that working with an *average project*[2] and deducing a priority list from this schedule, results in better expected makespan for the stochastic RCPSP than simply working with the reactive policies without using a baseline schedule at all.

In general terms, a baseline schedule that is rather 'insensitive' to disruptions that may occur during project execution is called *robust*. Many different types of robustness have been identified in the literature. We refer to Section 2.2 for a detailed overview. In this work, constructing a baseline schedule (instead of using a policy) that performs well on makespan or any due date related performance measure for a wide range of execution scenarios, corresponds to building a *quality robust schedule*, i.e. a schedule that is insensitive to the disruptions that affect the value of the performance metrics used to evaluate the quality of the schedule. Very few efforts have been made in order to construct such baseline schedules. The well-known critical chain buffer management approach of Goldratt (1997) might be considered as a quality robust baseline scheduling method because it adds buffers to

[2]The deterministic schedule deduced from the stochastic scheduling problem by setting the activity durations equal to its expected durations.

the schedule in order to ensure good makespan performance.

1.4 Stable project scheduling

All research efforts described in the previous section aim to minimize the expected time overrun of the project in a stochastic environment. However, the Standish Group survey also mentioned cost overrun as an important reason of project failure. Research on this topic is far less extensive. The time/cost trade-off problem (defined by Fulkerson (1961)) is a well-known cost minimizing problem that recognizes that the majority of activities can be performed in shorter durations by increasing the resources available to them. Mostly this increased resource allocation comes at a higher cost.

In real-life stochastic project settings, not only activity crashing but also delaying the activity behind its planned schedule start time, might induce a cost. Constant rescheduling in order to improve the expected makespan, might strongly decrease the predictive value of the baseline schedule. Project schedules should also include some *solution robustness* to cope with the uncertainties during project execution such that the *realized project schedule*, i.e. the list of actually realized activity start times during project execution, will not differ too much from the original baseline schedule.

One possibility to maximize solution robustness is to include safety in the baseline schedule in order to absorb the anticipated disruptions as well as possible. This is called *proactive scheduling*. A second approach, *reactive scheduling*, consists of defining a procedure to react to disruptions that cannot be absorbed by the baseline schedule.

Note that a pure proactive scheduling is an utopia, incorporating safety in a baseline schedule that allows to cope with every possible disruption would lead to a baseline schedule with a very large makespan. Every baseline scheduling method needs to be accompanied by a reactive procedure, but the number of interventions by the reactive procedure will be inversely related to the degree of proactiveness included in the baseline schedule.

In this dissertation, we concentrate on the development of efficient proactive-reactive scheduling procedures to introduce solution as well as quality robustness in a stochastic project environment. In the next chapter,

notations and definitions will be provided to formally describe the problem that will be tackled in the remaining chapters.

Chapter 2

Definitions and problem formulation

In this chapter a definition of the proactive-reactive project scheduling problem will be given. Section 2.1 introduces project representations that can assist us in illustrating the procedures that will be proposed in later chapters. Afterwards, we propose a rigorous definition of the concepts *quality* and *solution robustness*, which both will be main issues throughout this thesis (Section 2.2). Section 2.3 describes the activity weights that are used in the stability cost function and Section 2.4 gives an overview of the proactive-reactive scheduling problem discussed in this thesis.

2.1 Project representations

A project consists of a number of *events* and *activities* or *tasks* that have to be performed in accordance with a set of precedence and resource constraints. Each activity has a duration. The deterministic expected duration of activity j will be expressed as d_j, while in an uncertain scheduling environment, the stochastic activity durations will be denoted by $\mathbf{d}_j$.

In order to illustrate the working principles of a number of proactive-reactive scheduling procedures, we introduce a project that will be used as our vehicle of analysis throughout the entire thesis. Table 2.1 provides an overview of the data. The project consists of 10 activities (activity 0 and

Table 2.1: Data for the problem instance

j	d_j	w_j	r_j	$successors_j$
0	0	0	0	1, 2, 3
1	4	2	5	4, 7
2	5	7	3	4
3	2	4	4	5
4	4	5	4	6
5	5	3	3	8
6	4	7	5	9
7	2	5	3	9
8	2	5	6	9
9	0	38	0	/

activity 9 are dummy activities representing the project start and finish) and is subject to finish-start, zero-lag precedence constraints and a single renewable resource constraint (see Section 2.1.3). The single renewable resource is assumed to have a constant per period availability a of 10 units. Expected activity durations d_j, activity weights w_j (see Section 2.3) and resource requirements (see Section 2.1.3) are also given. In the remainder of this section we offer an overview of several possible project representations.

2.1.1 Project network

The main focus of the project network is the representation of the precedence relationships between the activities of the project. Precedence relationships may be of different types. We restrict our research to the best-known type of precedence relationships (PERT, CPM and RCPSP are also restricted to them), i.e. the finish-start relationships with zero time lag. The impact of Generalized Precedence Relationships (Elmaghraby & Kamburowski 1992) on proactive-reactive scheduling lies outside the scope of this dissertation.

A project network is a graphical representation of the events, activities

and precedence relations of the project. A network is a directed graph $G = (N, A)$, consisting of a set of nodes N and a set of arcs A. The *transitive closure* of a graph $G = (N, A)$ is a graph $TG = (N, TA)$ which contains an edge from i to j whenever there is a directed path from i to j in the original graph.

There are two network notation schemes commonly used in project scheduling. The *activity-on-arc* (AoA) representation uses the set of nodes N to represent events and the set of arcs A to represent the activities, while in the *activity-on-node* (AoN) notation scheme the set N denotes the activities and the set A represents the precedence relationships. The arcs TA of the transitive closure $TG = (N, TA)$ represent in this case all direct and transitive precedence relationships in the original network. In the remainder of this thesis we will opt for the AoN network representation. Figure 2.1 denotes the AoN project network for the project described in Table 2.1.

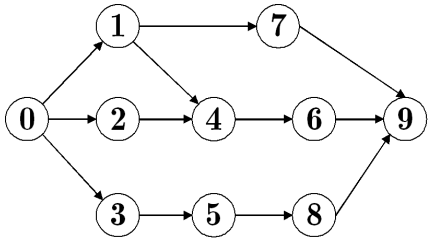


Figure 2.1: Problem network instance

2.1.2 Project schedules

A *schedule* S is defined in project scheduling as a list $S = (s_0, s_1, \ldots, s_n)$ of intended start times $s_j \geq 0$ for all activities $j \in N$. A Gantt chart (introduced by H. Gantt in 1910) provides a typical graphical schedule representation by drawing the activities on a time axis.

A schedule is called *feasible* if the assigned activity start times respect the constraints imposed on the problem. In deterministic project scheduling, a feasible schedule is a sufficient representation of a solution. Figure 2.2 depicts a solution schedule for our example network presented in Table 2.1. It will be shown in Chapter 5 that this schedule corresponds to the optimal

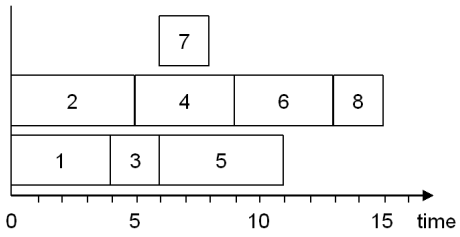


Figure 2.2: A minimum duration schedule

RCPSP schedule found by the branch-and-bound code of Demeulemeester & Herroelen (1992, 1997).

In stochastic scheduling a schedule needs to be accompanied by a reactive policy or strategy (see Chapter 7) to compose a solution. However, this does not reduce the importance of schedules as visualization tools for project management. Many types of schedules are relevant in proactive-reactive project scheduling.

2.1.2.1 Baseline schedule

A *baseline schedule* (*pre-schedule or predictive schedule*) is a list of activity start times generated under the assumption of a static and deterministic environment that is used as a baseline during actual project execution. A baseline schedule is generated before the actual start of the project (time 0) and will consequently be referred to as S^0. It serves a number of important functions (Aytug et al. (2005), Mehta & Uzsoy (1998), Wu et al. (1993)) One of them is to provide visibility within the organization of the time windows that are reserved for executing activities in order to reflect the requirements for the key staff, equipment and other resources. The baseline schedule is also the starting point for communication and coordination with external entities in the company's inbound and outbound supply chain: it constitutes the basis for agreements with suppliers and subcontractors (e.g. for planning external activities such as material procurement and preventive maintenance), as well as for commitments to customers (delivery dates). An important decision to be made in the proactive-reactive scheduling problem is the amount of safety to include in the baseline schedule. Chapters 4, 5

and 6 are completely devoted to this decision.

2.1.2.2 Initial schedule

As will be seen later in this chapter, we mostly adopt a two-stage approach that starts building a stable baseline schedule from an unprotected initial schedule. This *initial schedule* or *unbuffered schedule* S^U will mostly either be:

- an already existing schedule in the organization;
- a schedule obtained by a commercial software package;
- a schedule generated by one of the procedures from the extensive deterministic RCPSP literature.

The fact that initial schedule selection might have a substantial impact on solution robustness, will be further examined in Chapter 6.

2.1.2.3 Realized schedule

A *realized schedule* S^T is a list of actually realized activity start times s^T that is generated once complete information of the project is gained. The proactive-reactive scheduling decisions made during project execution influence the actually obtained realized schedule S^T. In a stochastic environment, the realized schedule will thus typically be unknown before the project completion time T. We will refer to this stochastic schedule by $\mathbf{S}^T$.

2.1.2.4 Projected schedule

A *projected schedule* as introduced by Leach (2000) in the CC/BM methodology (see Section 3.2) is available at every moment in time and reflects a prediction of how the project scheduler expects the realized schedule will look like. As a project executes in an uncertain environment, at each decision time t $(0 < t < T)$ gradually more information about the project becomes available, which will lead to a constant updating of the projected schedule S^t.

2.1.2.5 Ex-post schedule

An *ex-post schedule* is defined as the best schedule that could have been obtained for a given quality measure if full information about the realized activity durations would have been available in the project planning phase. It is unrealistic to expect that the realized schedule equals this ex-post schedule. However, it might be interesting to construct the ex-post schedule in order to compare the obtained objective function value of the realized schedule with the utopian objective function value that would have been obtained if full information had already been available in the project planning phase. The ex-post schedule will be indicated as S^*.

2.1.3 Resource usage and representations

In resource-constrained project scheduling, project activities require resources to guarantee their execution. Multiple resource categories exist (Blazewicz et al. 1986). In this thesis (as in the RCPSP), we will limit our scope to *renewable resources* that are available on a period-by-period basis and for which only the total resource use in each time period is constrained for each resource type. Every activity j requires an integer per period amount r_{jk} of one or more renewable resource types k ($k = 1, 2, ..., K$) during its execution. The renewable resources have a constant per period availability a_k. The resource constraints can thus be written as:

$$\sum_{\forall j \in P_t} r_{jk} \leq a_k \qquad \forall k = 1, 2, ..., K \qquad \forall t = 1, \ldots, s_n \tag{2.1}$$

in which P_t denotes the set of activities that are active at time t.

The network (see Section 2.1.1) and schedule (see Section 2.1.2) representations of the project do not visualize the resource allocation. Hence, additional resource-based project representations are introduced in this section.

2.1.3.1 Resource profile

A *resource profile* is an extension of a Gantt chart that additionally indicates the variation in resource requirement of a single renewable resource

type over time for each activity. Resource requirements and availability are denoted on the Y-axis. Each resource type requires its own resource profile. The resource profile for the single renewable resource type corresponding to the minimum duration schedule of Figure 2.2 is given in Figure 2.3.

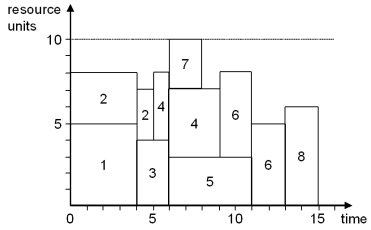


Figure 2.3: Resource profile for example project

2.1.3.2 Resource flow network

Artigues et al. (2003) introduce *resource flow networks* (or *transportation networks*) to identify the amount of resources transported from the end of one activity to the beginning of another activity after scheduling has taken place. f_{ijk} denotes the amount of resources of type k, flowing from activity i to activity j. The resource flow network is a network with the same nodes N as the original project network, but with arcs connecting two nodes if there is a resource flow between the corresponding activities, i.e. $\exists k : f_{ijk} > 0$. We define R as the set of flow carrying arcs in the resource flow network. The resource arcs in R may induce extra temporal constraints to the project. Policella et al. (2004) introduce the similar concept of a *partial order schedule* to represent the extra temporal constraints required to solve the resource allocation problem.

Remark that a schedule may allow for different ways of allocating the resources so that the same schedule may give rise to different resource flow networks. Not every feasible resource allocation implies an equal amount of stability. Solving the resource allocation problem for stability will be the topic of Chapter 4.

Relying on the one-pass algorithm of Artigues et al. (2003) to compose a resource flow network for the schedule of Figure 2.2, results in the resource

flow network $G = (N, R)$ presented in Figure 2.4. Activity 8, for example, has a per period resource requirement of six units. It uses three resource units released by its predecessor activity 5, two units passed on by activity 7 and one unit released by activity 6. The arcs (1,3); (3,7); (6,8), (7,6) and (7,8) represent extra precedence relations that were not present in the original network. The arc (7,9) was present in A, but is not drawn in Figure 2.4 because there is no resource flow from 7 to 9.

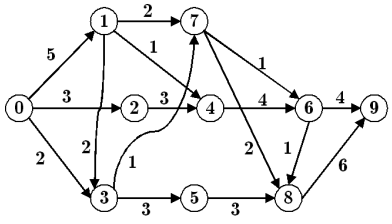


Figure 2.4: Resource flow network for the example project

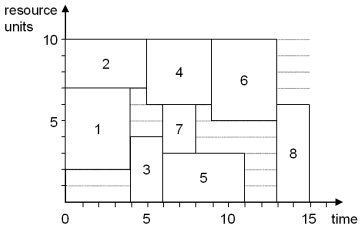


Figure 2.5: Resource profile with resource allocation

In Figure 2.5 we have redrawn the resource profile of Figure 2.3 to illustrate the use of the individual resource units along the horizontal bands. This project representation includes all information present in schedule, resource profile and resource flow network and will thus become our preferred representation in the remainder of this dissertation if there is only one resource type, as is the case in the example network. Only the precedence relationships of the original network $G = (N, A)$ are not explicitly represented in Figure 2.5.

2.2 Robustness types and measures

In Chapter 1, we already introduced the concept of schedule robustness, i.e. a schedule's insensitivity to disruptions that may occur during project execution. The robustness concept has been used in many disciplines (see e.g. Kouvelis & Yu (1997), Roy (2002), Billaut et al. (2005)). Many different types of robustness have been identified in the literature, calling for rigorous robustness definitions and the use of proper robustness measures.

A robustness measure can be single or composite. Two often used types of *single robustness measures* have been distinguished: *solution* and *quality robustness* (Sörensen (2001), Herroelen & Leus (2005)). The main difference between quality robustness and solution robustness is that in the former case, it is the quality of the solution that is not allowed to change. This quality is usually measured in terms of makespan or due date performance. In the latter case, it is the solution itself that is not allowed to change. For other typologies we refer to Sanlaville (2004).

2.2.1 Solution robustness or schedule stability

Solution robustness or schedule stability refers to the *difference* between the baseline schedule and the realized schedule. The difference or *distance* $\Delta(S^0, S^T)$ between the baseline schedule S^0 and the realized schedule S^T for a given execution scenario can be measured in a number of ways: the number of disrupted activities, the difference between the planned and realized activity start times, etc.

For example, the difference can be measured by the weighted sum of the absolute deviation between the planned and realized activity start times, i.e.

$$\Delta(S^0, S^T) = \sum_j w_j \left| s_j^0 - s_j^T \right|, \tag{2.2}$$

where s_j^0 denotes the planned starting time of activity j in the baseline schedule S^0, s_j^T denotes the actual starting time of activity j in the realized schedule S^T, and the weights w_j represent the disruption cost of activity j per time unit, i.e. the non-negative cost per unit time overrun or underrun on

the start time of activity j (see Section 2.3). In a stochastic environment, the realized activity starting times are stochastic variables $\mathbf{s}_j^T$ for which the actual realized values s_j^T for a given execution scenario depend on the disruptions and the applied reactive policy. The objective of the proactive-reactive scheduling procedure is then to minimize

$$\sum_j w_j E\left|s_j^0 - \mathbf{s}_j^T\right|, \tag{2.3}$$

with E denoting the expectation operator, i.e. to minimize the weighted sum of the expected absolute difference between the planned and the realized activity start times. It should be observed that the analytic evaluation of this objective is very cumbersome, the PERT problem being $\#P$-complete (Hagstrom 1988)[1]. Mostly this objective function will be evaluated by simulation. The obtained objective function values will then be dependent on the simulated disruptions and the applied reactive procedure (Leon et al. 1994).

Sanlaville (2004) suggests to measure solution robustness as

$$\max_I \Delta(S^0, S^T), \tag{2.4}$$

the maximum difference between the baseline schedule S^0 and the realized schedule S^T over the set of execution scenarios I. The objective of the proactive-reactive scheduling procedure then is to minimize this maximum distance.

Remark that the above solution robustness measures rely on knowledge of the realized schedule $\mathbf{S}^T$ and are thus very hard to evaluate. Monte Carlo simulation or approximation schemes (for overviews see Ludwig et al. (2001), Herroelen & Leus (2005) and Dodin (2006)) for the distribution functions of the activity starting times are required to approximate the stochastic starting times $\mathbf{s}_j^T$ in $\mathbf{S}^T$. In the following, some objective functions are introduced that do not require these computationally costly techniques to evaluate the

[1]Hagstrom in fact proved that calculating the probability of a single point in the cumulative distribution function of $\mathbf{s}_n$ is $\#P$ complete in the case of two-point distributions of the activity durations. It is generally assumed that these results can be extended to more general distribution functions of the activity duration such that efficient procedures for calculating the expected project duration are very unlikely to exist. This result authorizes simulative evaluation of the objective functions instead of analytical evaluation.

solution robustness. We will call them *surrogate objective functions* or *predictive objective functions*.

Schwindt (2005) uses the concepts of *early free float* (EFF) and *late free float* (LFF) to calculate solution robustness. EFF_j and LFF_j are defined as follows for the schedule S with starting times s_j, precedence relationships A and resource flow network $G(N, R)$:

$$EFF_j = s_j - \max_{(i,j)\in(A\cup R)}(s_i + d_i) \qquad \forall j \in N \tag{2.5}$$

$$LFF_j = \min_{(j,i)\in(A\cup R)} s_i - (s_j + d_j) \qquad \forall j \in N \tag{2.6}$$

He then introduces a solution robustness metric that maximizes the total weighted free float:

$$\max \sum_{j\in N} w_j(EFF_j + LFF_j) \tag{2.7}$$

in which w_i can be chosen to reflect the degree of uncertainty with respect to start time s_i.

Rather than working with free floats, Leus (2003) suggests to base surrogate objective functions for stability on *pairwise floats* PF_{ij}:

$$PF_{ij} = max(0, s_j - (s_i + d_i)) \qquad \forall i \in N; \forall j \in N. \tag{2.8}$$

$MSPF_{ij}$ is then defined as the minimal sum of pairwise floats between all subsequent activities on a path from activity i to activity j. If $(i, j) \notin T(A \cup R)$, $MSPF_{ij} = 0$.

Hence, he proposes the objective function $min \sum_{(i,j)\in TA} w_j PF_{ij}$ to evaluate several buffer insertion techniques for a given graph $G(N, A \cup R)$. In Deblaere et al. (2006), a solution robust resource allocation R for a given schedule S is found by maximizing:

$$\sum_{\forall(i,j):s_i+d_i\leq s_j} MSPF_{ij}, \tag{2.9}$$

where they set $MSPF_{ij}$ equal to a constant if there exists no path between i and j. A different surrogate objective function based on $MSPF_{ij}$ will be introduced in Chapter 5 of this dissertation (Eq. 5.6).

Lambrechts et al. (2006a) suggests an objective function that is based on diminishing costs per extra unit of free slack. They suggest to maximize the following solution robustness measure:

$$\sum_{i=1}^{n} CIW_i \sum_{j=1}^{FF_i} e^{-j}, \tag{2.10}$$

where FF_i denotes the (late) free float of activity i and CIW_i is the cumulative instability weight, which is calculated by adding w_i to the sum of the weights of all direct and indirect successors of i. The idea behind this objective function is that the necessity to buffer an activity diminishes as the activity is already buffered.

Policella et al. (2004) aim to obtain a solution robust resource allocation by minimizing the deviation in several evaluation criteria between the original project graph $G = (N, A)$ and the partial order schedule $G = (N, A \cup R)$ obtained after resource allocation. The *fluidity* metric is taken from Cesta et al. (1998) and defined as follows:

$$fldt = 100 \times \sum_{i \neq j} \frac{slack_{ij}}{H \times n \times (n-1)}, \tag{2.11}$$

where H is the predefined horizon of the problem, n is the number of activities and $slack_{ij}$ is the width of the allowed distance interval between the late finish time of activity i and the early start time of activity j. The hope is that the higher the value of *fldt*, the less the risk of a domino effect, and the higher the probability of localized changes. The *flexibility* (*flex*) measure is taken from Aloulou & Portmann (2003) and counts the number of pairs of activities that are not precedence related in $G(N, A \cup R)$. The rationale for this measure is that when two activities are not precedence related, it is possible to shift one of them in time without moving the other one. Hence, the higher the value of *flex*, the lower the degree of interaction among the activities and this will eventually lead to improved solution robustness. The work of Policella et al. (2004) will be discussed in more detail when the resource allocation problem is tackled in Chapter 4.

2.2.2 Quality robustness

Quality robustness refers to the insensitivity of some deterministic objective value of the baseline schedule to distortions. The goal is to generate a solution for which the objective function value does not deteriorate when disruptions occur. Contrary to solution robustness, quality robustness is not concerned with the solution itself, only with its value on the performance metric. It is measured in terms of the value of some objective function $\mathbf{z}$. In a project setting, commonly used objective functions are project duration (makespan), project earliness and tardiness, project cost, net present value, etc.

When stochastic data are available, quality robustness can be measured by considering the *expected value of the objective function*, such as the expected makespan $E[\mathbf{C}_{\max}]$, the classical objective function used in stochastic resource-constrained project scheduling (Stork 2001).

It is logical to use the *service level* as a quality robustness measure, i.e. to maximize $P(\mathbf{z} \leq z)$, the probability that the objective function value of the realized schedule stays within a certain threshold z. For the makespan objective, we want to maximize the probability that the project completion time does not exceed the project due date δ_n, i.e. $P(\mathbf{s}_n^T \leq \delta_n)$, where $\mathbf{s}_n^T$ is a stochastic variable that denotes the starting time of the dummy end activity in the realized schedule. We will refer to this measure as the *timely project completion probability* (TPCP). It should be observed that also the analytic evaluation of this measure is very troublesome in the presence of ample resource availabilities.

Quality robustness can also be measured by comparing the solution value $\mathbf{z}$ of the realized schedule obtained by the proactive-reactive scheduling procedure and the optimal solution value $\mathbf{z}^*$ computed ex-post by applying an exact procedure on the basis of the realized activity durations. Herroelen & Leus (2001), for example, have used the percentage deviation of $\mathbf{C}_{\max}$, the project duration of the realized schedule, from the ex-post optimal makespan $\mathbf{C}_{\max}^*$ computed by applying a branch-and-bound procedure on the basis of the actually realized activity durations, as a measure of quality robustness.

Al-Fawzan & Haouari (2005) consider the objective of quality robustness maximization by stating that a schedule is more robust if the total

sum of free slacks over all its activities is larger. Free slack ensures that disruptions will not be propagated to later stages of the schedule, which will improve the realized project makespan.

2.2.3 Composite robustness measures

The robustness measures described above are all single measures. It is also possible to use composite objectives. For extensive overviews of multicriteria scheduling, we refer the reader to T'kindt & Billaut (2006) and Hoogeveen (2005). We are interested in the bi-criteria objective of maximizing the timely project completion probability and minimizing the weighted sum of the expected absolute deviation in activity starting times:

$$F(P(\mathbf{s}_n^T \leq \delta_n), \sum w_j E\left|\mathbf{s}_j^T - s_j^0\right|) \tag{2.12}$$

We assume that the composite objective function $F(.,.)$ is not a priori known and that the relative importance of the two criteria is not known in the initial schedule development phase, i.e. the decision maker has no knowledge of e.g. a linear combination that reflects his preference.

2.3 Activity weights

The weighted sum of the absolute deviations between the planned and realized activity start times has been suggested as a solution robustness measure in previous sections. This section devotes attention to these activity dependent weights which play a central role in this thesis.

Once a project schedule has been negotiated, constructed and announced to all stakeholders, modifying this schedule comes at a certain penalty cost. This cost corresponds to the importance of on-time performance of a task to avoid *internal* and *external stability* costs. Internal stability costs for the organization may include unforeseen storage costs, extra organizational costs or just a cost that expresses the dissatisfaction of employees with schedule nervousness. They can also reflect the difficulty in shifting the booked time window on the required resources. Costs related to (renegotiating) agreements with subcontractors, penalty clauses, goodwill

damages, etc. are examples of stability costs that are external to the organization. In practice, these penalties may be considerable. For example, the penalty for not meeting the delivery date of the renovated Berlaymont building, housing the European Commission in Brussels, was set to 221,000 € per month of delay (European Commission 2004).

The activity-dependent weights w_j used in this dissertation represent the marginal cost of starting the activity j later or earlier than planned in the baseline schedule. The assumptions that earliness costs equal tardiness costs and that these costs have constant marginal returns per extra unit could easily be relaxed. Because most procedures proposed in this dissertation heavily rely on simulation to evaluate the objective function, these relaxations would not change the validity of our research.

The start of the dummy end activity of a project network represents the project completion time. We assume that the project will not be penalized for completing earlier than its predefined project due date δ_n. The weight of the dummy end activity w_n identifies the cost of delivering the project later than planned and will typically be higher than the cost of rescheduling an intermediate activity. The ratio $wp = w_n/w_{avg}$ between the weight of the dummy end activity and the average of all other activity weights is called the *weighting parameter, wp*. This wp will later be shown to be the driving force of the trade-off between stability and makespan.

2.4 Problem definition and organization of the thesis

In the previous sections of this chapter, several definitions and assumptions for the proactive-reactive scheduling problem were introduced. Table 2.2 gives a concise overview of the most important assumptions that will be made in this dissertation. In the remainder of this section, we will situate the proactive-reactive project scheduling problem discussed in this thesis by positioning it on the project life cycle (see Demeulemeester & Herroelen (2002)).

The aim of this thesis is to propose procedures for making a project with renewable resources and start-finish precedence relationships solution

Table 2.2: Overview of the assumptions made in this dissertation

Objective function	• Minimize $\sum_j w_j E\left\lvert s_j^0 - \mathbf{s}_j^T\right\rvert$ • Marginal earliness costs and marginal lateness costs are constant and equal
Activities	• Activity durations are stochastic • Multiple disruptions are possible • Realized activity durations are only known at the activities' completion time • Activity preemption is not allowed • Activities have a single execution mode
Precedence relationships	• No generalized precedence relationships
Resource usage	• Resource requirements and availabilities are deterministic and constant over time • Only renewable resources
Scheduling	• Activity starting times are integer • Project due date is predefined
Reactive	• On schedule breakage, we try to repair the baseline schedule in the best possible way • Activities are not allowed to start earlier than planned (except in Chapter 7)

robust without neglecting the importance of quality robustness. Many decisions have to be made during the project life cycle that can influence the quality of the project. In Figure 2.6 the proactive-reactive project planning decisions are situated on the project life cycle. We limit our elucidation of the project life cycle to the relevant concepts for our research. We refer to Demeulemeester & Herroelen (2002) for comprehensive phase definitions.

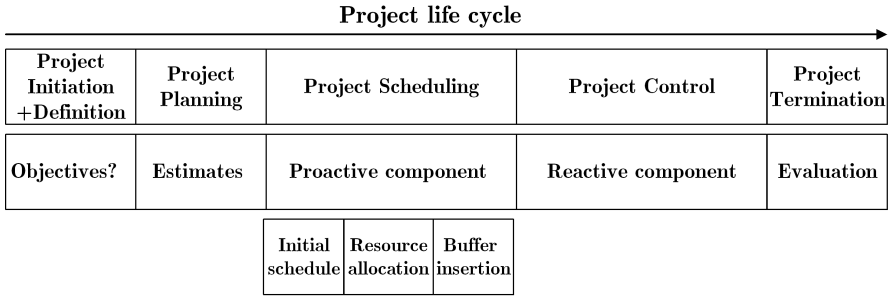


Figure 2.6: Proactive-reactive scheduling decisions to be made over the project life cycle

During *project initiation* an organization identifies the need for a project. In the next phase (*project definition*), a clear definition of the exact expectations of the project has to be formulated. Already in this early phase, a very important decision has to be made for the proactive-reactive project planning: the organization has to decide upon the objectives of the project. Also, the scope of the project has to be defined.

The *project planning* phase identifies the project activities, temporal relationships, resource usages, etc. Crucial to this phase for the problem discussed in this thesis is the need for careful estimates of expected durations, variations of durations, activity weights, etc. We must stress that the quality of the efforts made in subsequent phases of the project life cycle is highly dependent on the quality of these estimates. In Chapter 9 will be explained how risk detection can be tackled in real-life projects.

Once the project and its components have been properly defined, the actual project scheduling can take place. The work in this thesis is mainly located in this phase. The desired output of the project scheduling phase is a workable *baseline schedule* (see Section 2.1.2), which can guide the organization during the execution of the project. We concentrate on making

this baseline schedule quality and solution robust, such that it will be little sensitive to unanticipated disruptions. During the scheduling phase several decisions have to be made that may influence the robustness of the project. All these decisions influence the robustness of the baseline schedule and are inevitably interrelated. Integrating them into one overall robust scheduling approach is the topic of Chapter 6, but this might become computationally too troublesome to be acceptable for commercial software packages and real-life projects. In this thesis, a two-stage approach will be applied. In the first stage, an initial resource and precedence feasible schedule S^U is generated without looking at robustness. In the second stage, this initial schedule will be transformed into a more robust baseline schedule S^0. The rationale for relying on such a two-stage approach is that we want to allow for the possibility to generate solution robust schedules starting from an existing precedence and resource feasible but unprotected schedule (e.g. the currently available project schedule in the organization or a schedule generated using any exact or heuristic solution for the RCPSP) without having to resolve the resource allocation problem. Adding solution robustness can be done in several ways. The robustness of a predictive schedule can be improved by defining a clever way to perform the *resource allocation* among the activities (Chapter 4). Inserting idle time (*buffer allocation*) into the previously unbuffered schedule can also improve the robustness and will be the main concern of Chapter 5.

The *control phase* embodies the implementation of the project during project execution. Progress must be constantly monitored and measured. Uncertainty leads to unforeseen events. Every proactive baseline schedule has to be accompanied by a reactive procedure (Chapter 7) that defines how to update the projected schedule whenever a disruption occurs that could not be absorbed. The reactive scheduling policy is along with initial scheduling, resource allocation and buffer allocation, the fourth factor that will influence the ultimate quality of the solution.

Project termination is the last phase of the project life cycle. Here we will evaluate whether the objectives have been met, whether our proactive and reactive decisions have been satisfactory, etc. Customer satisfaction is the ultimate aim of every project.

Chapter 3

Trade-off between stability and makespan

During the last decade a lot of research efforts in the project scheduling literature have concentrated on resource-constrained project scheduling under uncertainty. Most of this research focuses on protecting the project makespan against disruptions during execution. Few efforts have been made to protect the starting times of intermediate activities. In this chapter, we address the issue whether to concentrate safety time in order to protect the planned project completion time (*quality robustness*) or to scatter safety time throughout the baseline schedule in order to enhance stability (*solution robustness*).

The chapter is organized as follows. In Section 3.1 we present a simple heuristic procedure for generating stable resource-constrained baseline schedules and provide illustration on the example problem introduced in Section 2.1.1. In Section 3.2 we exploit the same problem instance to describe our implementation of the critical chain (CC/BM) scheduling methodology for generating so-called buffered baseline schedules and unbuffered projected schedules. The experimental results are described in Section 3.3. A last section presents some overall conclusions.

3.1 A heuristic procedure for generating solution robust schedules

Herroelen & Leus (2004) and Leus (2003) describe a heuristic procedure for generating buffered baseline schedules for projects with ample renewable resource availability. Basically, their *adapted float factor heuristic* (ADFF) is an adaptation of the float factor model which was originally introduced by Tavares et al. (1998) to generate a schedule S in which the start time of activity i is obtained as

$$s_i(S) := s_i(ESS) + \alpha(s_i(LSS) - s_i(ESS)) \tag{3.1}$$

where $\alpha \in [0, 1]$ is the so-called *float factor*, $s_i(ESS)$ denotes the earliest possible start time of activity i and $s_i(LSS)$ represents the latest allowable start time of activity i. Both start times are derived from critical path calculations for a given project due date. Instead of using a single float factor α for all the activities, ADFF adopts an *activity dependent float factor* which is calculated as

$$\alpha_i = \beta_i/(\beta_i + \lambda_i) \tag{3.2}$$

where β_i is the sum of the weight of activity i and the weights of all its transitive predecessors, while λ_i is the sum of the weights of all successors of activity i in the transitive closure TA. In doing so, ADFF inserts longer time buffers in front of activities that would incur a high cost if started later than originally planned.

Obviously, when applied to a resource-constrained project, ADFF scatters intermediate time buffers throughout a baseline schedule but does not prohibit resource conflicts from occurring because neither the early start schedule nor the late start schedule are guaranteed to be resource feasible. To ensure that the buffered baseline schedule is resource feasible, the ADFF procedure is modified as follows.

The first step is to obtain a good precedence and resource feasible starting schedule. A number of exact and metaheuristic procedures for generating minimum duration schedules for the RCPSP have been described in the literature (see Demeulemeester & Herroelen (2002)). For illustrative purposes, we use the branch-and bound procedure of Demeulemeester

& Herroelen (1992, 1997) for generating an unbuffered resource-constrained schedule S^U. The project network introduced in Chapter 2 (Table 2.1) will be our vehicle of analysis. The *initial unbuffered schedule* obtained by the branch-and-bound procedure is shown in Figure 2.2. The *critical sequence*, i.e. the precedence and resource-constrained chain of activities that determines the 15-period makespan is the chain $< 0, 2, 4, 6, 8, 9 >$. The project due date δ_n is set to 20, a 30% increase above the critical sequence length. Note that alternative initial schedules are possible. Figure 3.1 gives the *right-justified* schedule that identifies all the latest allowable starting times to obtain the project due date in the deterministic case. For every activity i, the float value $float(i)$ is calculated as the difference between its latest allowable starting time (its starting time in Figure 3.1) and its scheduled starting times s_j^U in the unbuffered initial schedule of Figure 2.2.

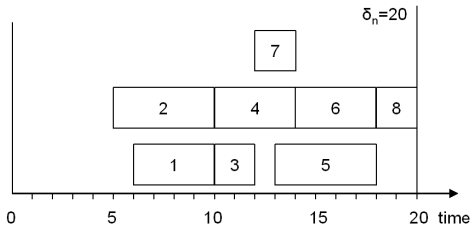


Figure 3.1: Right-justified schedule

In a second step the starting time of each activity i is calculated as

$$s_i(S) := s_i^U + \alpha_i float(i) \tag{3.3}$$

However, using the ADFF float factors $\alpha_i = \beta_i/(\beta_i + \lambda_i)$, does not ensure that the resulting $s_i(S)$ are resource feasible. Indeed, although both the activity starting times in the minimum duration schedule and in the right-justified schedule are resource feasible, this might not be the case for the computed $s_i(S)$. In order to obtain a precedence and resource feasible schedule, a set of different float factors α_i has to be used. For this purpose a *resource flow network* (see Section 2.1.3.2) is constructed. Multiple possible flow networks can be deduced. The next chapter of this thesis will be completely devoted to the importance of resource allocation in robust project scheduling. For now, we have opted for the single-pass procedure described

by Artigues et al. (2003) to generate a resource flow network that is feasible for each resource type.

The β_i's will now be calculated as the sum of w_i and the weights of all its predecessors in $TG = (N, T(A \cup R))$. Similarly, the (transitive) successors of activity i in both the original and the resource flow network are used to calculate the λ_i's. The weights of activities that start at time 0 are not included in these summations because it is assumed that these activities can always start at their planned start time and thus do not need any buffering to cope with possible disruptions of their predecessors. The resulting *resource flow dependent float factors* α_i consequently ensure that the RFDFF heuristic inserts longer time buffers in front of activities that would incur a high cost if started earlier or later than originally planned and that resource constraints will always remain satisfied in the resulting schedule.

Table 3.1 lists for each activity of the example project of Table 2.1 the values needed for the computation of the resource flow dependent float factor α_i and the scheduled starting time s_i. Note that we set $w_1 = w_2 = 0$ because activities 1 and 2 start at time 0. Activity 6 has activity 4 as its only direct predecessor in the project network of Figure 2.1 and activities 4 and 7 as direct predecessors in the resource flow network of Figure 2.4. Activities 0, 1, 2 and 3 are its transitive predecessors in these networks. This results in $\beta_6 = w_6 + w_3 + w_4 + w_7 = 21$. Similarly, summing the weights of all direct and transitive successors of activity 6 in the networks of Figure 2.1 and Figure 2.4 gives $\lambda_6 = w_8 + w_9 = 43$. We thus find $\alpha_6 = 21/(43+21) = 0.328$ and $s_6 = 9 + 0.328 \times 5 = 10.64$, which is discretized (rounded to the nearest integer) to 11. The starting times of the other activities can be calculated equivalently, resulting in the RFDFF schedule of Figure 3.2.

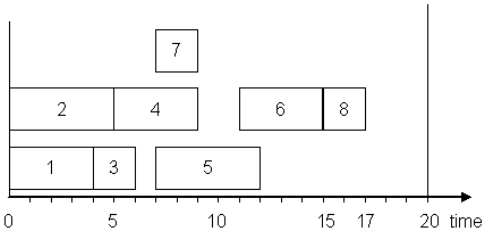


Figure 3.2: RFDFF schedule

Table 3.1: Values for the RFDFF heuristic

i	s_i^U	$float$(i)	w_i	β_i	λ_i	α_i	s_i
0	0	5	0	0	67	0	0
1	0	6	0	0	67	0	0
2	0	5	0	0	55	0	0
3	4	6	4	4	58	0.065	4
4	5	5	5	5	50	0.091	5
5	6	7	3	7	43	0.140	7
6	9	5	7	21	43	0.328	11
7	6	6	5	9	50	0.153	7
8	13	5	5	29	38	0.433	15
9	20	5	38	67	0	1	20

When an RFDFF schedule (or any other buffered baseline schedule) is executed, the included safety times have to be preserved in one way or another in order to preserve their stability advantage. The common project scheduling practice of starting activities as soon as possible during execution, is irreconcilable with buffer inclusion. Chapter 7 focuses on solution robust reactive procedures.

For now, buffers will be preserved by impeding activities to start earlier than their starting times s_j^0 in the baseline schedule S_j^0. This execution policy is commonly referred to as *railway scheduling*, because of its comparability with the scheduling of trains in a railway station, which states that a train is not allowed to leave the railway station earlier than planned because the time table holds commitments with travelers. In our example network, activity 5 for example will never start earlier than at time 7, even if its predecessors have already finished and all required resources are available.

3.2 Critical chain buffer management (CC/BM)

It has been advocated that project planning practitioners should rely on the well known *critical chain/buffer management* (CC/BM) methodol-

ogy (Goldratt 1997) - the direct application of the Theory of Constraints (TOC) to project management - to ensure that a project can be delivered on time in an uncertain project environment. In the experimental set-up of the next section, CC/BM will be used as the quality robust project scheduling method to examine the trade-off between makespan and stability. The fundamental working principles of CC/BM have been discussed by Goldratt (1997), Newbold (1998) and Herroelen & Leus (2001).

CC/BM builds a baseline schedule using aggressive median or average activity duration estimates. The safety in the durations of activities that was cut away by selecting aggressive duration estimates is concentrated at the end of the schedule in the form of a *project buffer* (PB) which should protect the project due date from variability in the critical chain activities. The *critical chain* is defined as the chain of precedence and resource dependent activities that determines the overall duration of a project. If multiple candidate critical chains exist, a random one is chosen. *Feeding buffers* (FB) are inserted whenever a non-critical chain activity joins the critical chain. This basically means that non-critical chains are pushed back in time. By doing this, new resource conflicts can be invoked. The literature is not that clear on how those conflicts should be solved. For executing a project, on the other hand, the CC/BM approach does not rely on the buffered schedule but on a so-called *projected schedule*. This schedule is precedence and resource feasible, contains no buffers and is to be executed according to the *roadrunner mentality*, i.e. the so-called *gating tasks* (activities with no non-dummy predecessors) are started at their scheduled start time in the buffered schedule while the other activities are started as soon as possible. The projected schedule is recomputed when disruptions occur. Neither the buffered schedule nor the projected schedule are constructed with a view to stability (*solution robustness*). In this section we will explain the implementation of CC/BM that we use in the remainder of this chapter.

First we solve the deterministic RCPSP by running the branch-and-bound code of Demeulemeester & Herroelen (1992, 1997). Because CC/BM starts with an as late as possible baseline schedule, we run the procedure on the inverse network and reverse the resulting schedule again to obtain a right-justified resource feasible unbuffered schedule. For our example network of

Table 2.1, this results in the schedule of Figure 3.3. The unique critical chain in this schedule is $< 0, 2, 4, 6, 8, 9 >$.

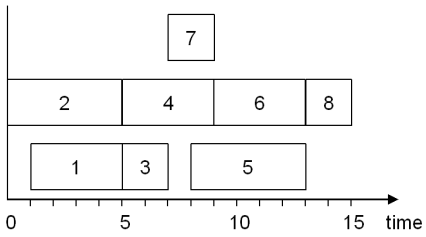


Figure 3.3: Right-justified schedule

Besides identifying the critical chain, we also have to compute the feeding buffers. For the example network in Figure 2.1, clearly three non-critical chains can be discovered. CC/BM adds feeding buffers between the last activity of a non-critical chain and the activity of the critical chain where this feeding chain joins the critical chain. In the example, we will add three feeding buffers, namely between the feeding chain $< 1 >$ and critical activity 4, between $< 3, 5 >$ and 8 and between $< 1, 7 >$ and 9. In this text, the size of a feeding buffer is set to 50% of the length of its feeding chain. For a detailed analysis of the impact of feeding buffer sizes on the trade-off between stability and makespan, the reader is referred to Van de Vonder et al. (2006b). Recently, Tukel et al. (2006) have described various more advanced methods to introduce feeding buffer sizes.

As has been demonstrated by Herroelen & Leus (2001), simply starting the feeding chains earlier in time to make room for the feeding buffers may introduce new resource conflicts. Instead of using some heuristic to resolve these resource conflicts, we opt for a complete rescheduling procedure (again by running the branch-and-bound code of Demeulemeester & Herroelen (1992, 1997)) in which the buffers are properly sized and considered as extra dummy activities with positive duration and no resource requirements, while assuring that the sequential order of the critical chain activities is kept unchanged. For the example network, this results in the buffered baseline schedule of Figure 3.4 where three feeding buffers and a project buffer of 50% of the critical chain length have been inserted.

For project execution, however, CC/BM does not rely on this buffered

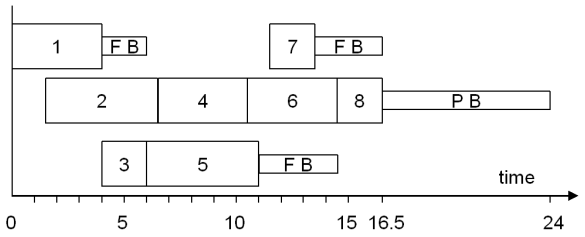


Figure 3.4: The buffered CC/BM baseline schedule

baseline schedule, but on the so-called *projected schedule*, which has been introduced earlier in this section. The alert reader will observe that the construction of such a projected schedule requires some additional information, which can for example be obtained by fixing the flows of a resource flow network. The derivation of the earliest possible activity starting times in the projected schedule not only depends on the original precedence constraints, but also on the resource flows between activities. All activities that pass on resources to other activities should be completed by the time these other activities start. Clearly, when disruptions occur during project execution, the projected schedule has to be recomputed. Figure 3.5 shows the initial projected schedule for our example network. Activity 7 is started earlier in time than in Figure 3.4, i.e. at the completion time of activity 5.

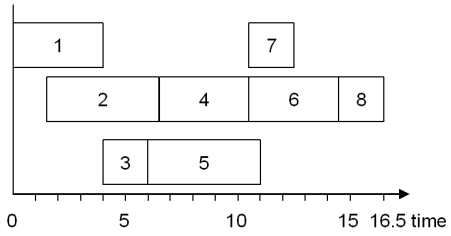


Figure 3.5: The initial CC/BM projected schedule

Figure 3.5 shows that CC/BM and RFDFF do not result in the same minimal project due date if implemented as described above. For RFDFF, a project buffer of 0% of project due date prolongation means that the project due date equals the critical chain length found by the RCPSP procedure. Contrarily, in CC/BM the critical chain does not necessarily start at time 0

because of the insertion of the feeding buffers. For example, in Figure 3.5, we remark that activity 2 only starts at time 1.5. This results in a project due date of 16.5 instead of 15, the value obtained by the RCPSP procedure of Figure 2.2. We will call this delay of 1.5 time units the *critical chain delay*. Thus, adding a zero-sized project buffer to the CC/BM schedule results in a makespan that equals the CC length plus CC delay. In order to obtain an honest comparison between both methods, we add the critical chain delay to the due date that is imposed on RFDFF to ensure that both methods have equal due dates.

3.3 Experimental results

We refer to Van de Vonder et al. (2006b) for an extensive computational experiment that aims to compare stability and makespan performance between a makespan protecting (i.e. quality robust) schedule and a stable or solution robust schedule. They set up a factorial design to examine the impact of buffer sizing decisions and of several parameter settings for project characteristics, such as the *order strength OS* (Mastor 1970), the *resource factor RF* (Pascoe 1966) and the *resource constrainedness RC* (Patterson 1976) on a set of 30-activity network instances generated by the RanGen project network generator of Demeulemeester et al. (2003).

In this text, we do not intend to revisit the detailed results of the factorial experiment, but rather limit our focus to the main conclusion drawn from that paper. It is intuitively clear that protecting for makespan performance, as done by CC/BM, will perform well on quality robustness measures such as TPCP. On the other hand, protecting individual activities for possible disruptions, as done by RFDFF, will certainly decrease the stability cost. The interesting issue addressed in this section, however, is the magnitude of the loss of makespan performance when protecting the intermediate milestones compared to the magnitude of the loss of stability when only protecting the makespan. In other words, we are interested in the performance of CC/BM and RFDFF on the composite robustness measure that was introduced in Eq. 2.12.

The importance of the weighting parameter wp should be emphasized.

This weighting parameter has been defined in Section 2.3 as the ratio between the weight of the dummy end activity and the average of the distribution of all other activity weights:

$$wp = \frac{w_n}{w_{avg}}. \tag{3.4}$$

Its impact on the stability objective measures is immense for both solution robust and quality robust baseline schedules. For RFDFF, the value of wp does not only affect the objective function values, but also the baseline schedules themselves (e.g. the schedule in Figure 3.2 assumed $wp = 10$). A higher wp means that the RFDFF procedure allocates a larger buffer to the dummy end activity. As such, a high wp improves the quality robustness of the solution robust schedule.

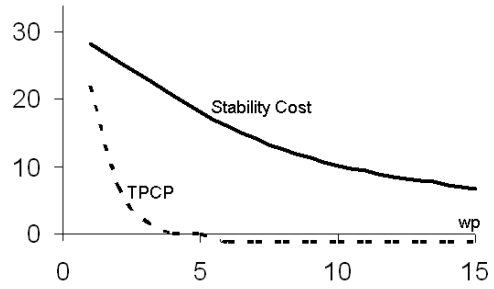


Figure 3.6: Comparing RFDFF and CC/BM for total buffering equal to 50% of CC length

Figure 3.6 summarizes the established trade-off. The bold curve indicates the ratio between the CC/BM stability cost for a 50% project due date delay compared to the RFDFF stability cost for the same due date. Obviously this advantage of RFDFF decreases for higher wp, but the difference remains substantial for every wp value considered. The dashed curve, on the other hand, denotes the difference in TPCP percentage points between CC/BM and RFDFF. This advantage of CC/BM in terms of makespan performance seems to decrease much more rapidly for increasing wp. We remark in Figure 3.6 that the dashed line even reaches negative values for large wp-values. In that case, the makespan advantage of CC/BM completely disappears because of the above described critical chain delay. For

real-life projects, low settings of wp are mostly unrealistic because the cost of not meeting the project due date will most probably greatly exceed the cost of not meeting an average planned activity starting time.

All of this leads to a paradoxical conclusion. The CC/BM philosophy tries to protect project completion because it assumes that project completion is much more important than the timely completion of intermediate activities (actually CC/BM rejects the use of milestones). However, Figure 3.6 shows that exactly when the weight w_n of the ending project activity is high, CC/BM becomes hard to defend. Even if we would make abstraction of the critical chain delay, we see that the huge advantage of RFDFF in stability cannot be compensated by the difference in makespan performance. The RFDFF procedure schedules activities in such a way that for high w_n values the resulting schedule is quality robust without compromising on solution robustness.

The above conclusions are very similar to the ones found in Van de Vonder et al. (2005c) for the non-resource-constrained case. Relaxing the resource constraints does lower stability cost, but almost equally for RFDFF and CC/BM. So, the stability cost ratio will hardly change. The difference in TPCP percentage points also shifts marginally between both cases, but not to the extent that it would change any conclusions.

3.4 Conclusions of the chapter

This chapter described a trade-off between makespan performance and stability, which is an important issue for every project. We have observed that the advantages of the two scheduling approaches developed in this chapter depend highly on the relative importance attributed to timely project completion compared to the importance attributed to timely completion of intermediate activities. The paradoxical fact that makespan protecting schedules (such as CC/BM) were shown to be hard to defend when makespan becomes very important, is an interesting conclusion. Improving project managers' awareness of the different scheduling strategies and their strengths and weaknesses is the ultimate aim.

We have drawn two important lessons from the CC/BM approach to

elaborate on in the remainder of this dissertation. First, buffer inclusion is able to substantially improve robustness. Second, project managers should schedule their project by using aggressive activity duration estimates (we assume them in the entire thesis), rather than directly including some amount of safety in these estimates. Coping to work with such tight activity durations is considered as a concern of communication and close project control. However, this chapter adds the comment that safety should not necessarily be grouped in a project buffer at the end of the project. Project characteristics and extensive risk analysis (see Chapter 9) should decide on both the position and the size of the buffer. More advanced heuristics for buffer insertion will be introduced in Chapter 5.

The examined trade-off between stability and makespan serves as an eye-opener for the importance of solution robust project scheduling. We proved that scheduling for stability does not necessarily invoke a lower makespan performance, while project (stability) costs can be largely reduced. The obtained results incite us to further develop algorithms that optimize the RCPSP for stability under uncertainty in the remainder of this thesis. In the next chapters, we will focus on the far less examined solution robustness rather than on quality robustness. Obtaining a satisfiable makespan performance will be achieved by developing solution robust procedures (such as RFDFF) that result in a schedule with sufficient attention for quality robustness. Estimates of w_n or wp are responsible for the quantification of the proportional importance that should be attributed to both robustness measures. The next chapters (4, 5 and 6) will concentrate on the proactive component, while Chapter 7 will focus on the reactive component of this problem.

Chapter 4

Solution robust resource allocation

Introducing robustness into a project schedule can be done in several ways. The robustness of a predictive schedule can be improved by defining a clever way in which the resources are allocated to the activities. In Section 2.1.3.2, a resource flow network has been defined as a graph $G(N, R)$ with the same set of nodes (N) as the original project network $G = (N, A)$, but resource arcs (R) are connecting two nodes i and j if there is a resource flow $f(i, j, k)$ of any resource type k between the corresponding activities. It denotes the way in which renewable resource units are transferred among the various project activities in the baseline schedule. However, a given baseline schedule may allow for different ways of allocating the resources so that the same schedule may be represented by different resource flow networks.

Resource allocation decisions may affect the robustness of the schedule in two ways. First, in our two-stage proactive scheduling approach (see Section 2.4), buffer insertion is always applied after resource allocation. This means that the resource flow network constructed on the initial schedule S^U remains valid for the buffered baseline schedule S^0. Second, as we will see in Chapter 7, a possible approach to react during project execution to disruptions that could not be absorbed by the proactive schedule, is to keep the resource flows intact. In doing so, the resource flows will dictate how

the schedule is repaired at each schedule breakage until a realized schedule S^T is obtained. Investing time to generate a robust resource allocation then becomes highly rewarding.

The outline of this chapter is as follows. In Section 4.1, we summarize the specialized literature on robust resource allocation. Next, we introduce a new approach to develop a robust resource allocation for a given schedule. In a last section some conclusions about the impact of the resource flows on robustness will be given. Computational results of our approach will be given in the integrated experimental design of Chapter 8. For an extensive overview and computational comparison of several resource allocation methods, we refer the reader to Deblaere et al. (2006).

4.1 Literature on robust resource allocation

For a given schedule, several possible resource allocations exist that differ in terms of robustness. Artigues et al. (2003) present a simple method for generating a feasible resource flow by extending a parallel schedule generation scheme to derive the flows. However, this procedure cannot be regarded as a robust resource allocation procedure and will only serve as benchmark algorithm.

Leus (2003) and Leus & Herroelen (2004) propose a branch-and-bound procedure that solves the *NP*-hard robust resource allocation problem for a single resource type. Constraint propagation is applied during the search to accelerate the algorithm. Extension to multiple resource types would require a revision of the branching decisions taken by the branch-and-bound procedure and the consistency tests involved in the constraint propagation.

Policella et al. (2004) and Policella (2005) introduce heuristic *chaining* procedures that are applied to transform an initial schedule into a *chained Partial Order Schedule* (POS). They define a *POS* as a set of solutions for the RCPSP that can be compactly represented by a temporal graph $G(N, A \cup R)$, which is an extension of the precedence graph $G(N, A)$, where N denotes the set of nodes (activities) and A denotes the precedence arcs. A set of additional resource arcs R is introduced to remove the so-called

minimal forbidden sets[1].

In total three chaining procedures are introduced by the authors, i.e. *basic chaining*, ISH and ISH^2. While in basic chaining the pairs of activities that are chained are basically picked randomly, ISH also starts by picking a random chain for one resource unit, but then gives priority to add an extra unit of resource flow to an arc (i, j) for which the activities i and j were already chained ($(i, j) \in R$). This reduces the number of resource suppliers in R for each activity $j \in N$. ISH^2 no longer starts by randomly picking a chain. Priority is given to chain activities that were already precedence related in the original project graph $G(N, A)$.

Recently, three new heuristics based on surrogate mixed integer programming (MIP) formulations of the strongly NP-hard resource allocation problem have been introduced by Deblaere et al. (2006). The $MinEA$ heuristic minimizes the number of extra precedence relations imposed by the resource allocation decisions, the $MaxPF$ heuristic maximizes the sum of pairwise floats (see Eq. 2.9) in the network $G(N, A \cup R)$ and the $MinED$ heuristic minimizes an approximation of the weighted stability cost. We remark that these procedures rely on a MIP solver such as ILOG's CPlex to solve the integer programming problems.

We refer to Policella et al. (2004) and Deblaere et al. (2006) for more details on the algorithms introduced in this section.

4.2 A constructive resource allocation algorithm

Our approach is quite different from the algorithms discussed in the previous section. Instead of using a surrogate objective function, we aim to minimize the non-stability cost $\sum_{j \in N} w_j E|s_j^T - s_j^0|$ by iteratively running for each activity the procedure MABO (*myopic activity based optimization*). The procedure is myopic because we do not look at other activities while deciding upon the best possible resource allocation for an activity. Unlike most existing resource allocation procedures, MABO also works rather

[1] A minimum forbidden set is defined as the minimal set of precedence unrelated activities which cannot be scheduled together due to the resource constraints (Igelmund & Rademacher 1983).

activity-based than resource-based. Before giving a detailed description of MABO (Section 4.2.2), we describe a preprocessing step (Section 4.2.1) that aims to reduce the complexity of the resource allocation problem through the identification of *unavoidable resource arcs*.

4.2.1 Problem complexity reduction

In this section, we aim to reduce the complexity of the resource allocation problem by introducing the concept of unavoidable resource arcs. Two activities i and j are connected by an unavoidable resource arc in the resource flow network for a given input schedule, if the schedule causes an unavoidable strict positive amount $f(i,j,k)$ of resource units of some resource type k to be sent from activity i to activity j. Defining $A_U \subset R$ as the set of unavoidable resource arcs in a feasible resource flow network $G = (N, R)$, the conditions to be satisfied by the activities i and j such that $(i,j) \in A_U$ can be formally specified as described in Theorem 1, in which P_{s_j} is the set of the activities that are in progress at time s_j and Z is the set of activities that have a baseline starting time s_z: $s_i + d_i \leq s_z < s_j$.

Theorem 1 $\forall i \in N; \forall j \in N$ *with* $s_j \geq s_i + d_i$:

$$(i,j) \in A_U \iff$$

$$\exists k : a_k - \sum_{l \in P_{s_j}} r_{lk} - max(0, r_{ik} - \sum_{z \in Z} r_{zk}) < r_{jk} \tag{4.1}$$

Proof At time s_j, the total amount of renewable resources of resource type k equals its per period availability a_k. The resource units that are currently allocated to the set of active activities P_{s_j} are unavailable for activity j. This results in $a_k - \sum_{l \in P_{s_j}} r_{lk}$ available resource items of type k at time s_j. These resource items are either currently allocated to activity i or to any other activity h with $s_h + d_h \leq s_j$. We know that at time $s_i + d_i$, r_{ik} resource items of type k were released by activity i and that only the activities in set Z, with an aggregated resource requirement of $\sum_{z \in Z} r_{zk}$, have been started

since and could have decreased the number of resources allocated to i. At least $r_{ik} - \sum_{z \in Z} r_{zk}$ remain allocated to activity i at time s_j. The maximum amount of resource units of type k that can be supplied to activity j at time s_j from other activities than activity i thus can be specified as in the left-hand side of Eq. 4.1. If this number is smaller than r_{jk}, there is an unavoidable resource flow between i and j.

The exact amount and resource type of the flows carried by the unavoidable resource arc are irrelevant for the time being. We are only interested in the fact that an arc (i, j) must be added to the set of unavoidable resource arcs A_U.

A similar approach has been proposed in the branch-and-bound procedure of Leus & Herroelen (2004), where in each node constraint propagation is used to update lower bounds LB_{ij} and upper bounds UB_{ij} of the flows between each pair of activities (i, j). If constraint propagation in the root node is able to detect an activity pair (i, j) with $LB_{ij} > 0$ then these activities are connected by an unavoidable resource arc.

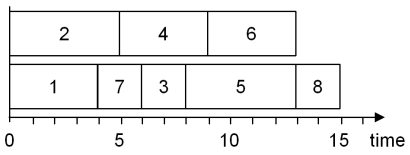


Figure 4.1: Minimum duration schedule

Consider our example project described in Section 2.1.1. Figure 4.1 is an alternative minimum duration schedule for this problem. We can show that this schedule requires an unavoidable resource arc from activity 7 to activity 3. At time $s_3 = 6$ only activity 4 is in progress with $r_4 = 4$. Because activity 3 starts when activity 7 ends, Z is obviously void. This results in the left-hand side of Eq. 4.1 being equal to $10 - 4 - max(0, 3 - 0) = 3$ which is smaller than $r_3 = 4$. $(7, 3)$ should thus be added to A_U. To investigate the existence of an unavoidable resource arc between activities 3 and 6, we again check Eq. 4.1. The left hand-side becomes: $a - r_5 - max(0, r_3 - r_5) = 10 - 3 - (4 - 3) = 6$ which exceeds the requirement of activity 6. A feasible resource allocation without a flow between activities 3 and 6 is thus possible.

The complete set of unavoidable resource arcs for the schedule of Figure 4.1 is $A_U = \{(0,1);(0,2);(1,7);(7,3);(3,5);(4,6);(6,8);(8,9)\}$. More important than this set A_U is the subset $A_U \setminus A$ of unavoidable resource arcs that do not correspond with a precedence arc in the original network $G(N,A)$. For our example schedule, we find $A_U \setminus A = \{(7,3);(6,8)\}$.

4.2.2 MABO

The MABO procedure consists of three steps which have to be executed for each activity $j \in N$. Step 1 examines whether the current predecessors of activity j have sufficient resource units available for j. If not, extra predecessors are added in a next step, while minimizing the impact on stability. Step 3 then defines resource flows $f(i,j,k)$ from predecessor activities i to activity j. The detailed steps of the procedure are depicted in the pseudo-code of Algorithm 4.1.

In the initialization step, the set of resource arcs R is initialized to the set of unavoidable arcs A_U. For each resource type k, the number of resource units $alloc_{0k}$ that may be transferred from the dummy start activity 0 is initialized to the resource availability a_k. The project activities are considered in increasing order of their planned starting times with decreasing weight as tie break rule.

In Step 1, we calculate the amount of resource units $avail_{jk}(A \cup R)$ currently allocated to the predecessors of activity j in $A \cup R$. If there exists any resource type k for which this amount of available resource units is not sufficient, new precedence constraints have to be added to R in Step 2. We define the set H_j of all possible arcs between a potential resource supplier h of the current activity j and j itself. By running a small recursion problem, we can find the subset H_j^* of H_j that accounts for the missing resource requirements of j for any resource type k at a minimum stability cost $\mathit{Stability\_cost}(A \cup R \cup H_j^*)$. This stability cost is the average stability cost $\sum_{j \in N} w_j E|\mathbf{s}_j - s_j|$, computed through simulation of 100 executions of the (partial) schedule, keeping the resource flows fixed (see Chapters 7 and 8), and respecting the additional precedence constraints $R \cup H_j^*$ that were not present in the original project network diagram. For alternative MABO

Algorithm 4.1 MABO

Initialize: $R = A_U$ and $\forall k : alloc_{0k} = a_k$

Sort the project activities by increasing s_j(tie break: decreasing w_j)

Take next activity j from list

1. Calculate $avail_{jk}(A \cup R) = \sum_{\forall i:(i,j)\in A\cup R} alloc_{ik}$ for each k

2. If $\exists k : avail_{jk}(A \cup R) < r_{jk}$

2.1 Define the set of arcs H_j

with $(h, j) \in H_j \iff$

$(h, j) \notin A \cup R$

$s_h + d_h \leq s_j$

$\exists k : alloc_{hk} > 0$

2.2 Find a subset H_j^* of H_j

such that $\forall k : avail_{jk}(A \cup R \cup H_j^*) >= r_{jk}$

and *Stability_cost*$(A \cup R \cup H_j^*)$ is minimized

2.3 Add H_j^* to R

3. Allocate resource flows $f(i, j, k)$ to the arcs $(i, j) \in (A \cup R)$:

For each resource type k:

3.1 Sort predecessors i of j by:

Increasing number of successors l of i

with $s_l > s_j$ and $r_{lk} > 0$

Tie-break 1: Decreasing finish times $s_i + d_i$

Tie-break 2: Decreasing variance σ_i^2 of $\mathbf{d}_i$

Exception: Activity 0 is always put last in the list

3.2 While $alloc_{jk} < r_{jk}$

Take next activity i from the list

$f(i, j, k) = \min(alloc_{ik}, r_{jk} - alloc_{jk})$

Add $f(i, j, k)$ to $alloc_{jk}$

Subtract $f(i, j, k)$ from $alloc_{ik}$

formulations, the stability cost function used in Step 2.2 can be replaced by a surrogate objective function (see Chapter 2) to reduce the computational requirements. However, it should be mentioned that more dedicated procedures for optimizing surrogate objective functions have been proposed in Deblaere et al. (2006).

The set of arcs H_j^* is added to R such that the updated $avail_{jk}(A \cup R) \geq r_{jk}$ and the resource allocation problem for the current activity is solved in a myopic way.

In Step 3, we allocate the actual resource flows $f(i, j, k)$ to the predecessors of j in $A \cup R$ and we update the number of resource $alloc_{ik}$ items allocated to each activity. Following Step 2 we know that $avail_{jk}(A \cup R) \geq r_{jk}$ for each activity j and each resource type k. If $avail_{jk}(A \cup R) = r_{jk}$ the available units are all allocated to the corresponding resource flows. If $avail_{jk}(A \cup R)$ is strictly larger than r_{jk} for resource type k, an excess of resource items of this type is available and we have to decide which predecessors will account for the resource flows and which are left vacant. We try to do this in an intelligent way, because a greedy algorithm would even reinforce the myopic character of MABO imposed in Step 2. The predecessors i are sorted by increasing number of their not yet started successors l with $r_{lk} > 0$, because these successors might count on these resources to be available. Two tie-break rules are used: decreasing finish times and decreasing activity duration variances. The idea is that the predecessors earlier in the sorted list normally have a higher probability to disrupt future activities. Having them as a resource supplier would thus in general be unfavorable for solution robustness. However, when these activities occur in the list of predecessors, it is advisable to consume as much as possible of the resource units they release such that their possible high impact on upcoming activities is neutralized. This allocation procedure is redone for every resource type k independently.

After all this we restart the three-step procedure for the next activity in the list until we have obtained a complete feasible resource allocation at the end of the list. The procedure uses an optimal recursion algorithm for each activity, but is not necessarily optimal over all activities.

As an illustration, we run MABO on the minimum duration schedule of Figure 4.1. Because the problem instance has a single resource type, we omit

the index k in further notations. We start by ordering the activities, yielding the list $(0, 2, 1, 7, 4, 3, 5, 6, 8, 9)$. All available resource units are allocated to the dummy start activity ($alloc_0 = 10$).

Activity 2 is next on the list. It has dummy activity 0 as single predecessor, so that $avail_2 = alloc_0 = 10$. As $avail_2 > r_2$ $(10 > 3)$, no extra precedence relations have to be added and we can proceed to Step 3. We set $f(0, 2) = \min(alloc_0, r_2 - alloc_2) = 3$, $alloc_0 = 7$ and $alloc_2 = 3$.

Also activity 1 poses no problems because its only predecessor (activity 0) still has 7 transferrable resource units and $r_1 = 5$. Thus, $f(0, 1) = 5$, $alloc_0 = 2$, $alloc_1 = 5$.

Activity 7 is the next activity on the list and we calculate in Step 1 that $avail_7((1, 7)) = alloc_1 = 5$, while $r_7 = 3$. Step 2 can thus again be skipped and the algorithm decides in Step 3 that $f(1, 7) = \min(alloc_1, r_7 - alloc_7) = 3$, $alloc_1 = 2$ and $alloc_7 = 3$.

Activity 4 is next. $avail_4((1, 4), (2, 4)) = 2 + 3 = 5$ while $r_4 = 4$. Step 3 gives priority to activity 2, because neither activity 1 nor activity 2 has any not started successors left, but activity 2 ends later in the baseline schedule and is thus a greater stability threat for activities further in the list. This results in $f(2, 4) = \min(alloc_2, r_4 - alloc_4) = 3$ and $alloc_4 = 3$ and $alloc_2 = 0$. Then $f(1, 4) = \min(alloc_1, r_4 - alloc_4) = 1$, $alloc_1 = 1$ and $alloc_4 = 4$. Activities 3 and 5 are processed in a similar way.

When activity 6 is looked at, the current situation is $alloc_0 = 1$, $alloc_1 = 1$, $alloc_3 = 1$, $alloc_4 = 4$ and $alloc_5 = 3$, resulting in $avail_6(4, 6) = alloc_4 = 4$, which is smaller than $r_6 = 5$. Thus, for the first time, an extra precedence relationship has to be added to supply 1 resource unit from $H_6 = \{(0, 6), (1, 6), (3, 6)\}$. Two subsets of H_6, namely $\{(0, 6)\}$ and $\{(1, 6)\}$ can resolve the resource allocation problem for activity 6 without extra cost. This is no surprise because both 0 and 1 are already transitive predecessors of 6. Activity 1 is selected to supply the missing resource unit and thus $(1, 6)$ is added to R. The procedure then moves on, until a complete feasible resource allocation is found.

Figure 4.2 shows the resource profile for the obtained robust resource allocation for the schedule of Figure 4.1. The horizontal bands define the resource flows between the activities. The random procedure of Artigues

et al. (2003) would result in the resource allocation illustrated in Figure 4.3. Obviously, especially the resource flow between activities 3 and 6 would account for an increased stability cost if the resource flows were preserved as reactive policy.

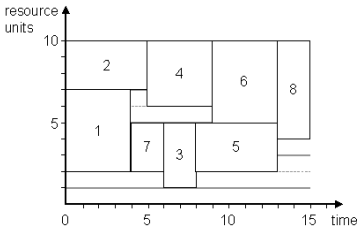


Figure 4.2: Robust resource allocation for schedule 4.1

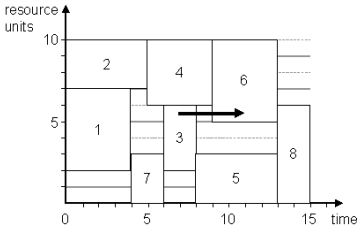


Figure 4.3: Random resource allocation for schedule 4.1

4.2.3 Lower bounds on schedule stability cost

In this section, we derive a lower bound on the schedule stability cost for a given schedule. Such a lower bound is independent of the resource allocation decisions and permits us to quantify the quality of the initial schedule S^U. Deriving a tight lower bound allows to evaluate the performance of the resource allocation procedures. We refer to Deblaere et al. (2006) for a computational analysis of the lower bounds of this section. For the subproblem with one renewable resource type, the performance of the resource allocation heuristics can be compared with the optimal results obtained by the branch-and-bound procedure of Leus & Herroelen (2004).

The lower bound calculations identify the stability cost contributions

that are indispensable. These include stability cost contributions due to the original precedence relations A and the unavoidable resource arcs A_U identified in Section 4.2.1. *Stability_cost*$(A \cup A_U)$ is thus a lower bound on the schedule stability cost that is independent of the resource allocation decisions. We will refer to this weak lower bound as LB_0.

Algorithm 4.2 presents a tighter lower bound which can be found by focusing on the resource allocation decisions that are not resolved by taking into account the unavoidable arcs $A \cup A_U$. We calculate for each activity j the best case scenario to solve the myopic resource allocation problem. We begin by calculating the minimal number of resource items $alloc_{ik}$ allocated at time s_j to each activity i with $s_i < s_j$ as $max(0, r_{ik} - \sum_{z \in Z_i} r_{zk})$. Z_i denotes again the set of activities that have a baseline starting time s_z: $s_i + d_i \leq s_z < s_j$. Summing up $\sum_{i \in N} alloc_{ik}$ might result in a total number of allocated resource units that is smaller than a_k. The difference $alloc_{xk} = a_k - \sum_{i \in N} alloc_{ik}$ represents the number of resource units of type k from unknown origin at the current time.

As in step 1 of the MABO procedure, we need to know the number of resource units that are allocated to predecessors of activity j in $A \cup A_R$. It is not sure whether the unknown origins of the $alloc_{xk}$ units are predecessors of activity j or not. In any case, there are no more than $avail_{jk} = alloc_{xk} + \sum_{\forall i:(i,j) \in A \cup A_R} alloc_{ik}$ resource units of type k allocated to predecessors of j at time s_j. If there exists a k for which $avail_{jk} < r_{jk}$, activity j must have non-predecessors as resource suppliers. Steps 2.1, 2.2 and 2.3 of MABO decide upon the best non-predecessors to supply extra resource units and calculate *Stability_cost*$(H_j^* \cup A \cup A_R)$. The current minimal allocations $alloc_{ik}$ are used as input.

After doing this for every activity, we identify the bottleneck activity j^* that is the most costly to resolve and calculate *Stability_cost*$(H_{j^*}^* \cup A \cup A_R)$. The so found stability cost is a tighter lower bound on the schedule stability cost. We will refer to this improved lower bound as LB_1.

During the calculation of the lower bound LB_1, we might encounter some activities for which the resource allocation problem cannot be solved

Algorithm 4.2 LB_1

$LB_1 \leftarrow \infty$
for each activity j with $s_j > 0$ **do**
  for each resource type k **do**
    for each activity i with $s_i < s_j$ **do**
      $alloc_{ik} = max(0, r_{ik} - \sum_{z \in Z_i} r_{zk})$
    $alloc_{xk} = a_k - \sum_{i \in N,\, s_i < s_j} alloc_{ik}$
    $avail_{jk}(A \cup A_R)^{\max} = alloc_{xk} + \sum_{\forall i:(i,j) \in A \cup A_R} alloc_{ik}$
  if $\exists k : avail_{jk}(A \cup A_R)^{\max} < r_{jk}$ **then**
    Define the set of arcs H_j with $(h,j) \in H_j \iff$
      $(h,j) \notin A \cup A_R$
      $s_h + d_h \leq s_j$
      $\exists k : alloc_{hk} > 0$
    Determine all minimal subsets $H_j^1, H_j^2, \ldots, H_j^m \subseteq H_j$ such that
      $\forall k : avail_{jk}(A \cup A_R \cup H_j^i)^{\max} \geq r_{jk}, \quad i = 1, \ldots, m$
    Identify the subset $H_j^* \in \{H_j^1, H_j^2, \ldots, H_j^m\}$ such that
      Stability_cost$(A \cup A_R \cup H_j^*)$ is minimized
    $LB_1 \leftarrow \min(\textit{Stability\_cost}(A \cup A_R \cup H_j^*), LB_1)$

Algorithm 4.3 LB_2

$I \leftarrow \emptyset$
$LB_0 \leftarrow$ *Stability_cost*$(A \cup A_R)$
Step 1:
for each activity j with $s_j > 0$ **do**
  for each resource type k **do**
    for each activity i with $s_i < s_j$ **do**
      $alloc_{ik} = max(0, r_{ik} - \sum_{z \in Z_i} r_{zk})$
    $alloc_{xk} = a_k - \sum_{i \in N,\, s_i < s_j} alloc_{ik}$
    $avail_{jk}(A \cup A_R)^{\max} = alloc_{xk} + \sum_{\forall i:(i,j) \in A \cup A_R} alloc_{ik}$
  if $\exists k : avail_{jk}(A \cup A_R)^{\max} < r_{jk}$ **then**
    Define the set of arcs H_j with $(h, j) \in H_j \iff$
      $(h, j) \notin A \cup A_R$
      $s_h + d_h \leq s_j$
      $\exists k : alloc_{hk} > 0$
    Determine all minimal subsets $H_j^1, H_j^2, \ldots, H_j^m \subseteq H_j$ such that
      $\forall k : avail_{jk}(A \cup A_R \cup H_j^i)^{\max} \geq r_{jk}, \quad i = 1, \ldots, m$
    Identify the subset $H_j^* \in \{H_j^1, H_j^2, \ldots, H_j^m\}$ such that
      Stability_cost$(A \cup A_R \cup H_j^*)$ is minimized
    if *Stability_cost*$(A \cup A_R \cup H_j^*) > LB_0$ **then**
      $I \leftarrow I \cup \{j\}$
      store $L_j = \{H_j^1, H_j^2, \ldots, H_j^m\}$
Step 2:
Given $I = \{j_1, j_2, \ldots, j_p\}$ with $p \geq 2$, identify the combination of subsets $H_{j_1}^* \in L_{j_1}, H_{j_2}^* \in L_{j_2}, \ldots, H_{j_p}^* \in L_{j_p}$ such that
  Stability_cost$(A \cup A_R \cup H_{j_1}^* \cup \ldots \cup H_{j_p}^*)$ is minimized
$LB_2 \leftarrow$ *Stability_cost*$(A \cup A_R \cup H_{j_1}^* \cup \ldots \cup H_{j_p}^*)$

without extra stability cost, i.e. $Stability\_cost(A \cup A_R \cup H_j^*) > LB_0$ for certain activities j. If this number of *stability cost increasing activities* is at least two, we can tighten the lower bound LB_1 even further, by looking at the combined effect of solving the resource allocation problem for each of these activities. The detailed steps of this tightened lower bound LB_2 are presented in Algorithm 4.3. The procedure consists of two steps. The first step is very similar to the calculation of LB_1: we identify all *stability cost increasing activities*, we add them to a set I and store the subsets of arcs able to solve their resource allocation problem. In a second step, we calculate the stability cost for all possible combinations of these subsets. As we are sure that one of these combinations of subsets will appear in an optimal resource flow network (w.r.t. schedule stability), the combination of subsets with minimal stability cost gives us a tightened lower bound.

4.3 Conclusions on resource allocation

In this chapter, we pointed out that resource allocation decisions might influence the solution robustness of a project baseline schedule. We have introduced a new approach to set up a robust resource flow network for a given schedule. We refer to Deblaere et al. (2006) for an extensive experimental analysis of several resource allocation methods and of the lower bounds proposed in this chapter. MABO shows to be among the best on PSPLIB J30 network instances and to be the very best for larger instance sizes. Its computational requirement even remains small (on average < 0.6 s on PSPLIB J120) for larger network sizes, while in that case the required computation time for the MIP formulations typically explodes exponentially.

In later chapters, MABO will be used as a robust resource allocation procedure and its effectiveness will be further analyzed in the computational experiment of Chapter 8. Fixing the resource allocation during project execution will be one of several reactive scheduling procedures discussed in Chapter 7. We will investigate in Chapter 8 the impact of using MABO compared to using the benchmark resource allocation procedure of Artigues et al. (2003) for this purpose.

Chapter 5

Solution robust buffer allocation

It has been demonstrated in Chapter 3 that when projects have to be executed in the face of uncertainty, proactive-reactive project scheduling procedures are capable of combining schedule stability and makespan performance. The use of an objective function aiming at schedule stability pays off. Several ways of enhancing the robustness of a schedule have been recognized. While the previous chapter dwelled on solution robust resource allocation, Leus (2003) also identified buffer insertion as a way to generate proactive project schedules.

The objective of this chapter is to develop and validate a number of heuristic and exact buffer insertion procedures for generating stable project baseline schedules. The algorithms described in this chapter all consider a deterministic project due date δ_n and start from an initial schedule S^U in which time buffers are inserted in order to protect against anticipated disruptions. Any feasible solution for the deterministic resource-constrained project scheduling problem (RCPSP) using mean activity durations (problem $m, 1|cpm|C_{\max}$ (Herroelen et al. 2000)) can serve as an initial unbuffered schedule. The impact of the choice of the initial unbuffered schedule on robustness will be examined in the next chapter. The included time buffers are idle periods (gaps) in the schedule between the latest planned finish time of the predecessors of an activity and the planned starting time of this ac-

tivity itself. The buffers should act as cushions to prevent propagation of a disruption throughout the schedule.

Before including buffers, the resource allocation problem is solved as explained in the previous chapter. The buffered baseline schedule S^0 will respect the resource allocation decision made on the unbuffered initial schedule S^U. This ensures that the buffered schedules that are generated remain resource feasible.

The chapter is organized as follows. Section 5.1 introduces the different heuristic and exact procedures. Section 5.2 presents the computational results, while a last section is devoted to overall conclusions.

5.1 Algorithms for solution robust buffer allocation

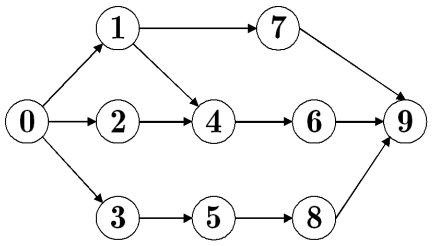


Figure 5.1: Problem network instance

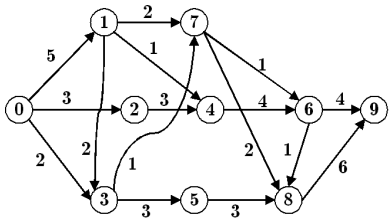


Figure 5.2: Resource flow network for example project

Including buffers into an initial schedule S^U in order to transform it into a solution robust baseline schedule S^0 is no easy task. The number of

possible schedules with integer activity starting times that can be built on a certain resource flow network is highly dependent on the project network size, the amount of slack present in S^U, the project network structure and the predefined project due date. For the example project network of Figure 5.1 (see also Figure 2.1), the number of feasible schedules that follow the resource allocation of Figure 5.2 (see also Figure 2.4) and have a due date that equals the minimum makespan s_n^U, can be easily verified as being equal to 8 if we assume that activities 1 and 2 always start at time 0. The schedules are shown in Figure 5.3, in which the shaded activities are those that start later than their starting time in the unbuffered schedule. It can be shown for this example that by increasing the due date by one time unit from $s_n^U + B - 1$ to $s_n^U + B$ time units, the possible number of additional schedules is:

$$\sum_{x=0}^{B}(x+1).\sum_{i=0}^{x+1}(4+B+x-2i)(i+1) \tag{5.1}$$

Proof Consider a full enumeration tree in which the branching levels correspond to the activities, considered in non-increasing order of their starting times in the unbuffered schedule S^U. For the example project, the resulting activity list is $(9, 8, 6, 7, 5, 4, 3, 2, 1, 0)$. In the root node (level 0), the dummy end activity is set equal to the due date.

Consider that the due date is increased by one time unit from $s_n^U + B - 1$ to $s_n^U + B$. This means that at level 1, one extra node arises that allows to start activity 8 at time $s_8^U + B$. This node creates a total of $B + 1$ new nodes at level 2. Indeed, given the decision made in level 1, activity 6 can now start at $s_6^U, \ldots, s_6^U + B$. The possible starting times for activity 7 at level 3, depend on the starting time of activity 6. If $s_6 = s_6^U + x$, $x + 2$ feasible starting times for activity 7 exist. This results in a total of $\sum_{x=0}^{x=B}(x+2)$ additional nodes at level 3. Each of these nodes leads to $B + 2$ nodes at level 4, because the possible starting times of activity 5 only depend on the starting time of activity 8, its sole predecessor in $G(N, A \cup R)$. Each of these $(B+2)\sum_{x=0}^{x=B}(x+2)$ nodes at level 4 corresponds to $x + 1$ nodes at level 5, in

which x equals $s_6 - s_6^U$. This gives us a total of $(B+2)\sum_{x=0}^{x=B}(x+1)(x+2)$ nodes at level 5. Enumerating the feasible starting times for activity 3 is more complex because the latest feasible finish time of activity 3 equals the minimum of the starting times of activities 5 and 7. The number of nodes at level 6 can be verified to equal $\sum_{x=0}^{B}(x+1).\sum_{i=0}^{x+1}(4+B+x-2i)(i+1)$. By assuming that activities 1 and 2 always start at time 0, this number also equals the number of nodes at branching levels 7 and 8 and by consequence equals the number of additional feasible schedules by increasing the project due date to $s_n^U + B$.

This means that for the project due date $\delta_n = 20$ already 4326 different schedules exist with $s_n \leq \delta_n$ that respect the resource allocation of Figure 5.2. For more complex and larger projects, the buffer insertion problem will obviously become far more complex.

It should be clear that enumerating and evaluating all candidate buffered schedules for a given resource flow network is unrealistic. Both heuristic and metaheuristic buffer allocation algorithms will be proposed, as well as an exact branch-and-bound procedure. Sections 5.1.1, 5.1.2 and 5.1.3 introduce simple heuristic procedures, while Sections 5.1.4 and 5.1.5 propose metaheuristics. Finally, an exact buffer allocation procedure is provided. They all aim to cleverly incorporate safety time into a schedule. Figure 5.4 illustrates a *buffer allocation* that makes a schedule proactive by including safety time in front of activities 5, 6 and 7. The project due date δ_n equals 20.

Two factors are taken into consideration in determining the size of the buffer in front of an activity i. First, the variability of all the activities that precede activity i in the schedule (measured by the standard deviation σ_j of their duration) is taken into account, because it affects the probability that activity i can start at its scheduled starting time. Second, the weight of activity i, and both the weights of its predecessors and successors contain relevant information, because they reflect how costly it is to disrupt the starting time of activity i in relation to its predecessors and successors in the schedule.

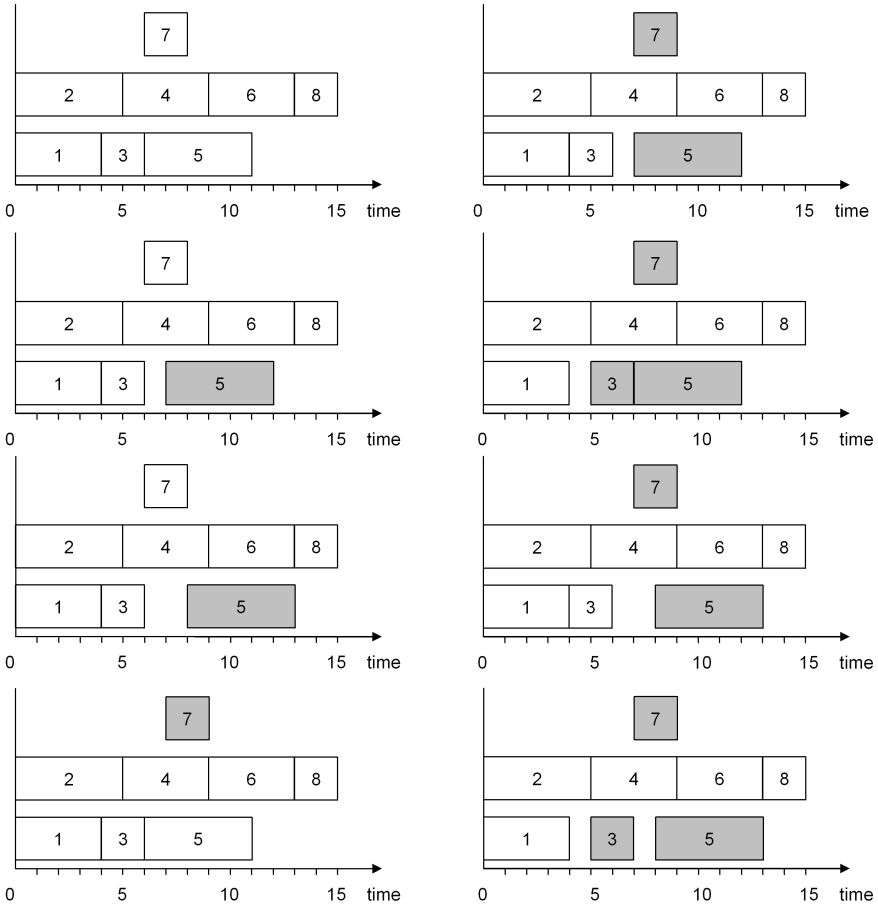


Figure 5.3: All feasible schedules with $s_n = \delta_n = 15$

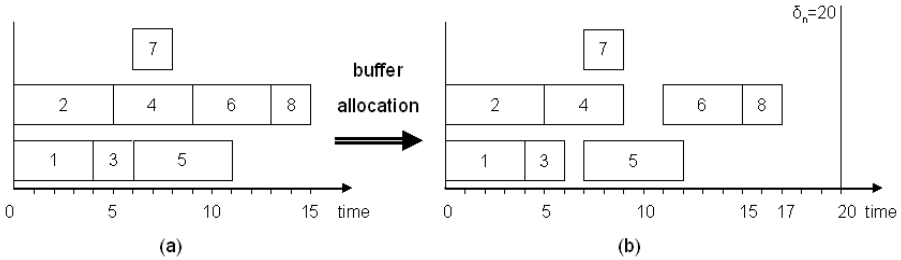


Figure 5.4: Inserting time buffers in a baseline schedule

Remark that during project execution, activities will never be allowed to start earlier than planned in order to preserve the stability advantage of the idle times in the schedule. This execution policy is commonly referred to as *railway scheduling* as already described in Section 3.1.

5.1.1 RFDFF

The resource flow dependent float factor model proposed in Section 3.1 is a buffer allocation heuristic that relies completely on the activity weights and does not exploit the available information offered by the activity duration distributions in making its buffering decisions.

We recall that the starting time of activity j in the RFDFF schedule is calculated as

$$s_j(S) := s_j^U + \alpha_j \times \mathit{float}(j), \tag{5.2}$$

where s_j^U is the starting time of j in the initial schedule and α_j denotes the *activity dependent float factor*, and is dependent on the weights of the predecessors and successors of activity j in $TG = (N, T(A \cup R))$, i.e. the transitive closures of the union of the original project network and the resource flow network. We refer to Section 3.1 for a more thorough description of this heuristic. Figures 3.2 and 5.4(b) denote the RFDFF schedule for our example project network of Figure 5.1.

5.1.2 VADE

The *virtual activity duration extension* (VADE) heuristic starts from a different point of view. The standard deviations σ_j of the activity durations, assumed known, are used to iteratively compute virtual duration extensions for the non-dummy activities. These virtual activity durations are used to update the activity start times and, by doing so, insert time gaps in the baseline schedule. The updated activity starting times are then used to generate the buffered baseline schedule using the original expected activity durations.

The iterative procedure works as described in the pseudo-code of Algorithm 5.1. Initially each non-dummy activity duration d'_j for $j = 1, 2, ..., n-1$ is set equal to its expected value $E(\mathbf{d}_j)$ and all $v_j = 1$, where v_j counts

Algorithm 5.1 VADE

For $j = 1, 2, ..., n-1$ do $d'_j = E(\mathbf{d}_j)$ and $v_j = 1$;
Compute s'_j, $j = 1, 2, ..., n$;
While $s'_n \leq \delta_n$ do
    Find $j^* : \frac{v_{j^*}}{\sigma_{j^*}} = \min_j \left\{ \frac{v_j}{\sigma_j} \right\}$
    (tie-break: $\max sw_j = \sum_{i=succ(j)} w_i$);
    $v_{j^*} = v_{j^*} + 1$;
    $d'_{j^*} = d'_{j^*} + 1$;
    Compute S' and s'_n;
Generate the buffered baseline schedule

the number of times that the duration of activity j has been virtually extended. The activity starting times s'_j are initially computed by creating an early start schedule for $G(N, A \cup R)$ using the activity durations d'_j. As long as the project duration stays within the due date δ_n, the activity start times are iteratively updated as follows. Determine the activity j^* for which $\frac{v_{j^*}}{\sigma_{j^*}} = \min_j \left\{ \frac{v_j}{\sigma_j} \right\}$. Ties are broken by selecting the activity for which the sum of the weights of all its non-dummy successors is the largest. Set $v_{j^*} = v_{j^*} + 1$ and $d'_{j^*} = d'_{j^*} + 1$. Afterwards, $\forall (j^*, k) \in (A \cup R)$: if $s'_{j^*} + d'_{j^*} > s'_k$ then the schedule needs to be updated by postponing s'_k by 1 time unit. This approach will be repeated for the successors of k until a resource and precedence feasible updated schedule S' is obtained. That schedule will be the input of the next iteration step. At every iteration step, S' will be evaluated by simulation and the schedule that minimizes $\Sigma w_j(\mathbf{s}_j - s'_j)$ will be the output schedule of the VADE procedure.

The standard deviations of the activity durations of our project example are given in Table 5.1. As $\frac{v_4}{\sigma_4} = 1.19 = \min_j \left\{ \frac{v_j}{\sigma_j} \right\}$, $d'_4 = 4 + 1 = 5$ and $v_4 = 1 + 1 = 2$, so that $\frac{v_4}{\sigma_4} = 2.38$. The virtual duration extension of activity 4 generates a one-period delay in the starting times of its successor activities 6 and 8 in the resource flow network, so that $s'_6 = 10$ and $s'_8 = 14$. Next, $\frac{v_5}{\sigma_5} = 1.59 = \min_j \left\{ \frac{v_j}{\sigma_j} \right\}$ so that $d'_5 = 5 + 1 = 6$, $v_5 = 1 + 1 = 2$ and $\frac{v_5}{\sigma_5} = 3.17$. The current schedule does not need to be modified because of the two-period gap between the completion time of activity 5 and the starting time of activity 8. Now we have that $\frac{v_7}{\sigma_7} = 1.69 = \min_j \left\{ \frac{v_j}{\sigma_j} \right\}$, $d'_7 = 2+1 = 3$

Table 5.1: Values for the VADE heuristic

j	$E(\mathbf{d}_j)$	σ_j	sw_j
1	4	0.56	14
2	5	0.31	5
3	2	0.40	8
4	4	0.84	7
5	5	0.63	5
6	4	0.28	5
7	2	0.59	12
8	2	0.20	0

and $v_7 = 1 + 1 = 2$, so that $\frac{v_7}{\sigma_7} = 3.39$. Activity 7 has a two-period float, so there is no need to update the schedule. Now $\frac{v_1}{\sigma_1} = 1.79 = \min_j \left\{\frac{v_j}{\sigma_j}\right\}$, $d'_1 = 5$, $v_1 = 2$ and $\frac{v_1}{\sigma_1} = 3.57$. Continuing the procedure in this manner leads to the virtual duration updates $d'_4 = 6$, $d'_3 = 3$, $d'_2 = 6$, $d'_5 = 7$, and $d'_7 = 4$. Now we have $\frac{v_1}{\sigma_1} = \frac{v_4}{\sigma_4} = \frac{v_6}{\sigma_6} = 3.57 = \min_j \left\{\frac{v_j}{\sigma_j}\right\}$. Invoking the tie-break rule (see column sw_j in Table 5.1) leads to the update $d'_1 = 6$. Subsequently, $d'_4 = 7$ and $d'_6 = 5$. However, this very last activity duration prolongation does not show any improvement when the objective function is evaluated by simulation. The current solution will thus not be stored as best so far, but will still be executed in order to continue the search. At this juncture, updating $d'_4 = 8$ would move the expected project duration beyond the project due date. The algorithm terminates and yields the schedule of Figure 5.5.

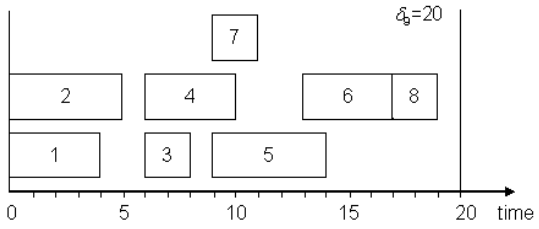


Figure 5.5: VADE schedule

5.1.3 STC

The *starting time criticality heuristic* (STC) exploits information about both the weights of the activities and the variance structure of the activity durations. The basic idea is to start from an unbuffered schedule S^U and iteratively create intermediate schedules by inserting a one-time period buffer in front of the activity that is the most starting time critical in the current intermediate schedule, until adding more safety would no longer improve stability. The starting time criticality of an activity j is defined as

$$stc(j) = P(\mathbf{s}_j > s_j) \times w_j = \gamma_j \times w_j \tag{5.3}$$

where γ_j denotes the probability that activity j cannot be started at its scheduled starting time.

The iterative procedure runs as follows. At each iteration step (see pseudocode below) the buffer sizes of the current intermediate schedule are updated. The activities are listed in decreasing order of the $stc(j)$. The list is scanned and the size of the buffer to be placed in front of the currently selected activity from the list is augmented by one time period such that the starting times of the activity itself and of the direct and transitive successors of the activity in $G(N, A \cup R)$ are increased by one time unit. If this new schedule has a feasible project completion ($s_n < \delta_n$) and results in a lower approximated stability cost ($\sum_{j \in N} stc(j)$), the schedule serves as the input schedule for the next iteration step. If not, the next activity in the list is considered. Whenever we reach an activity j for which $stc(j) = 0$ (all activities j with $s_j = 0$ are by definition in this case) and no feasible improvement is found, a local optimum is obtained and the procedure terminates.

Regrettably, the probabilities γ_j are not easy to compute. We define $k(i, j)$ as the event that predecessor i disturbs the planned starting time of activity j. The probability that this event occurs can be expressed as $P(k(i, j)) = P(\mathbf{s}_i + \mathbf{d}_i + LPL(i, j) > s_j)$ in which $LPL(i, j)$ is the sum of the durations d_h of all activities h on the longest path between activity i and activity j in the graph $G(N, A \cup R)$. γ_j can then be calculated as $\gamma_j = P(\bigcup_i k(i, j))$ for $\forall i : (i, j) \in T(A \cup R)$. STC makes two assumptions in approximating γ_i: (a) predecessor i of activity j starts at its originally planned starting time when calculating $P(k(i, j))$ and (b) only one activity

Algorithm 5.2 The iteration step of the STC heuristic

```
Calculate all stc(j)
Sort activities by decreasing stc(j)
While no improvement found do
    take next activity j from list
    if stc(j)=0: procedure terminates
    else add buffer in front of j
        update schedule
        if improvement & feasible do
            store schedule
            goto next iteration step
        else
            remove buffer in front of j
            restore schedule
```

at a time disturbs the starting time of activity j. Assumption (b) means that we estimate $P(\bigcup_i k(i,j))$ by $\sum_i P(k(i,j))$, i.e. we assume that $P(k(i1,j) \cap k(i2,j)) = 0$ for each $i1, i2$. Assumption (a) boils down to setting $\mathbf{s}_i = s_i$. Combining both assumptions yields

$$\gamma'_j = \sum_i P(\mathbf{d}_i > s_j - s_i - LPL(i,j)) \tag{5.4}$$

such that $stc(j) = \gamma'_j \times w_j$. Because s_i, s_j and $LPL(i,j)$ and the distribution of $\mathbf{d}_i$ are all known, we can now easily calculate the values of γ'_j and $stc(j)$ for every activity j. This results in the measure for stability that is depicted in Eq. 5.5.

$$\min \sum_{j \in N} stc(j) = \min \sum_{\forall j \in N} w_j \sum_{\forall i:(i,j) \in T(A \cup R)} P(\mathbf{d}_i > s_j - s_i - LPL(i,j)) \tag{5.5}$$

Besides its repetitive use in STC, this expression can also be used as a surrogate objective function for the objective of Eq. 2.3. For that purpose, Eq. 5.5 can be rewritten as Eq. 5.6 where the closely related minimum sum of

Table 5.2: The longest path lengths from i to j

$LPL(i,j)$	1	2	3	4	5	6	7	8	9
1			0	0	2	4	2	8	10
2				0		4		8	10
3					0	2	0	6	8
4						0		4	6
5								0	2
6								0	2
7						0		4	6
8									0

pairwise floats $MSPF_{ij}$ concept (see Chapter 2) is used instead of $LPL(i,j)$[1].

$$\min \sum_{j \in N} stc(j) = \min \sum_{\forall j \in N} w_j \sum_{\forall i:(i,j) \in T(A \cup R)} P(\mathbf{d}_i > d_i + MSPF_{i,j}) \quad (5.6)$$

The application of STC to the problem example of Table 2.1 and the schedule of Figure 2.2 runs as follows. First, the $LPL(i,j)$ values need to be calculated for all predecessors i for every activity j. For illustrative purposes, we calculate $LPL(1,3)$, $LPL(1,5)$ and $LPL(1,8)$. A glance at the minimum duration schedule of Figure 2.2 and the resource flow network of Figure 5.2 reveals that activity 3 is immediately preceded by activity 1 and thus $LPL(1,3) = 0$. The resource flow network of Figure 5.2 shows a unique path $< 1,3,5 >$ leading from activity 1 to activity 5. This results in $LPL(1,5) = E(\mathbf{d}_3) = 2$. Multiple paths exist between activity 1 and 8, namely $< 1,3,5,8 >$, $< 1,3,7,8 >$, $< 1,3,7,6,8 >$, $< 1,7,8 >$, $< 1,7,6,8 >$ and $< 1,4,6,8 >$ with a corresponding path length of 7, 4, 8, 4, 2, 6 and 8, respectively. Thus, $LPL(1,8) = 8$. Table 5.2 shows all $LPL(i,j)$-values. If $(i,j) \notin T(A \cup R)$, the corresponding cell in the table is left blank.

The *stc*-values are first calculated for the initial minimum duration schedule of Figure 2.2 with $s_n = \delta_n = 20$. For example $stc(6)$ is calculated

[1]The advantage of using LPL in STC is that these values only need to be calculated once in the iterative STC procedure because they are independent of activity starting times.

as $w_6 \times (P(k(1,6)) + P(k(2,6)) + P(k(3,6)) + P(k(4,6)) + P(k(7,6)))$ with
$P(k(1,6)) = P(\mathbf{d}_1 > s_6 - s_1 - LPL(1,6)) = P(\mathbf{d}_1 > 5) = 0.11$
$P(k(2,6)) = P(\mathbf{d}_2 > s_6 - s_2 - LPL(2,6)) = P(\mathbf{d}_2 > 5) = 0.23$
$P(k(3,6)) = P(\mathbf{d}_3 > s_6 - s_3 - LPL(3,6)) = P(\mathbf{d}_3 > 3) = 0.01$
$P(k(4,6)) = P(\mathbf{d}_4 > s_6 - s_4 - LPL(4,6)) = P(\mathbf{d}_4 > 4) = 0.34$
$P(k(7,6)) = P(\mathbf{d}_7 > s_6 - s_7 - LPL(7,6)) = P(\mathbf{d}_7 > 3) = 0.05$
This results in $stc(6) = 7 \times (0.11 + 0.23 + 0.01 + 0.34 + 0.05) = 5.18$. All the *stc*-values for the starting solution are shown in column *Initial* in Table 5.3. $\sum stc(i) = 16.93$ denotes the total cost of the schedule and provides a good measure for the stability of the schedule. Ordering the activities by decreasing *stc* gives (**6**, 8, 7, 4, 5, 3, 9, 1, 2). Adding a one-time period buffer in front of activity 6 yields updates of the starting times $s_6 = 10$ and $s_8 = 14$ and provides the input schedule for Step 1.

Table 5.3: Computational steps of the STC procedure

Act	*Initial*	*Step* 1	*Step* 2	*Step* 3	*Step* 4	*Step* 5	*Step* 6	*Step* 7
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	1.20	1.20	1.20	1.20	1.20	1.20	1.20	0.44
4	1.71	1.71	1.71	1.71	1.71	0.25	0.25	0.25
5	1.47	1.47	1.47	1.47	1.47	1.47	0.35	0.09
6	5.18	1.69	2.02	2.02	0.62	0.58	0.58	0.60
7	2.45	2.45	0.58	0.58	0.58	0.58	0.58	0.15
8	4.88	2.22	2.45	0.48	0.20	0.20	0.20	0.20
9	0.04	0.04	0.04	0.04	0.04	0.27	0.27	0.27
Tot	16.93	10.78	9.47	7.50	5.82	4.55	3.43	2.00

The newly inserted buffer in front of activity 6 requires a recalculation of its *stc*-value and the *stc*-value of its successors activities 8 and 9. However, observe that *stc*(9) does not change although the buffer between activities 8 and 9 is reduced in size by one time period. Indeed, activity 8 has a very low variability and $P(k(8,9)) = 0$, indicating that the buffer size between activity 8 and 9 does not need to be increased. Table 5.3 shows the

new values in the Step 1 column. Activity 7 now has the largest *stc*-value, yielding the ordered list (**7**, 8, 4, 6, 5, 3, 9, 1, 2). Delaying the starting time of activity 7 by one time period is feasible and leads to a reduction in the total schedule cost.

In Step 2, $stc(7)$ will decrease while $stc(6)$ and $stc(8)$ will increase because of the delay of s_7. Activity 8 has the largest *stc*-value and will be delayed by one time period in the input schedule of Step 3.

Similarly to Step 1, the buffer in front of activity 9 is large enough and thus only $stc(8)$ drops in value, resulting in the new ordered list (**6**, 4, 5, 3, 7, 8, 9, 1, 2). Inserting a second one-time period buffer in front of activity 6 will not cause any feasibility problems and will improve the total schedule stability cost.

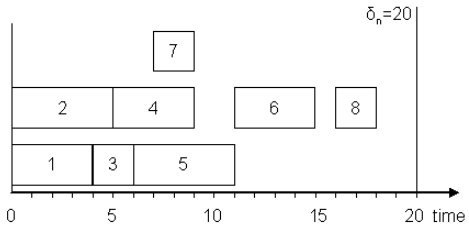


Figure 5.6: The input schedule for step 4

The input schedule for Step 4 is given in Figure 5.6. Obviously, the insertion of the buffer in front of activity 6 will result in a decrease of $stc(6)$. Also $stc(8)$ will slightly decrease. Activity now 4 has the highest *stc*- value and will thus be delayed.

For the input schedule of Step 5, we find that $s_4 = 6$. However, because the buffer size in front of activities 6 and 8 were previously fixed at respectively two and one time periods, also s_6 and s_8 will be delayed by one time period. This results in the positive side effect that also $stc(6)$ will decrease. The ordered list becomes (**5**, 3, 6, 7, 9, 4, 8, 1, 2) and the total schedule cost has decreased in value to 5.82.

In Step 6 the buffer size in front of activity 5 will be set to one time period. No other activity shift is required. Observe the difference with the previous step. Here the buffer size in front of activity 8 remains respected by delaying a predecessor, while in Step 5, s_8 had to be delayed in order to

respect the time period buffer. Table 5.3 again shows the updated *stc*-values. Activity 3 is now the most time critical and will be delayed.

Inserting a buffer in front of activity 3 results in a major shift in the schedule. The starting times of activities 5 and 7 will both be delayed by one time period in order to keep their previously allocated buffer sizes fixed. The corresponding *stc*-values will decrease and $\sum stc(i) = 2$. A small increase of $stc(6)$ is the only drawback. The *Step 7* column in Table 5.3 shows the newly ordered list (6, 3, 9, 4, 8, 7, 5, 1, 2). The algorithm continues by trying to delay activity 6. By doing so $stc(4)$ would decrease from 0.60 to 0.22 and $stc(8)$ would decrease from 0.20 to 0.07, but $stc(9)$ would explode by a stunning 2.17 resulting in a higher total schedule cost. Delaying activity 6 is thus infeasible and we proceed with the next activity in the list, namely activity 3. However, buffering activity 3 or either of the following activities in the list can also be proven to result in non-improving moves. The procedure terminates and Figure 5.7 represents the schedule found by the STC heuristic for our example project.

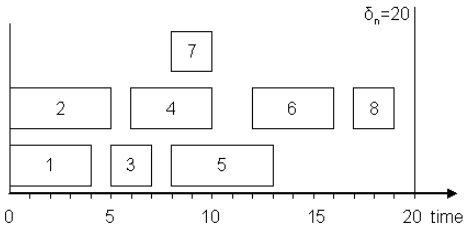


Figure 5.7: STC schedule

5.1.4 Improvement heuristic

The improvement heuristic starts from an initial solution. This can be the schedule found by any of the heuristics discussed above or the unbuffered schedule S^U. The activities are entered in a list in decreasing order of their starting time in the input schedule. The activities are considered in the order dictated by the list. For the currently selected activity from the list, it is determined how many periods the activity can be moved backward and forward in time without affecting the starting time of any other activ-

ity in the schedule. A neighborhood of the current solution is constructed by including for each discrete time instant in this displacement interval a schedule with s_j equal to this time and all other starting times remaining unchanged. These schedules are then all evaluated by simulation and the time instant that yields the lowest stability cost, is chosen as the new starting time of the current activity in the updated schedule. With this updated schedule, we proceed to the next activity in the list. If the next activity in the list is the dummy start activity, we restart the list. If the list is scanned entirely without any improvement, a local optimum has been found and the procedure terminates.

Basically the algorithm is a combination of steepest and fastest descent. For an activity selected from the list, we examine all possible starting times and select the best one (steepest descent). However, we do not examine the entire range of starting times of *all* the activities and select the best, but instead we already update the schedule if a better solution is found for the current activity before proceeding to the next activity in the list. This fastest descent part of the algorithm is included to speed up computations.

5.1.5 Tabu search

Descent approaches may terminate at a local optimum after some iterations when no further improvement can be found in the direct neighborhood of the current solution. Glover (1989, 1990) developed the principle of tabu search algorithms, which allow to select the mildest ascent solution to continue the search whenever no improvement can be found. A tabu list keeps track of recent solutions that will be forbidden moves in order to avoid cycling.

The tabu search procedure starts with the STC schedule described in Section 5.1.3. At each iteration step, the neighborhood of the current solution contains at most $2\times(n-2)$ solutions. For each non-dummy activity of the project, we have two possible neighborhood solutions. One is obtained by increasing the buffer in front of the activity in the schedule by one time period, if possible (*plus-move*). The other is obtained by decreasing the buffer size of this activity by one unit, if possible (*minus-move*). The buffers in front of all other activities are left unchanged. Two tabu lists are kept,

both of length $\lfloor n/3 \rfloor$. The first list stores all recent plus-moves, while the second one stores all recent minus-moves. Before allowing a new plus-move, we have to check whether this activity is not in the second list. If a buffer size decrease (minus-move) delivers the best solution in the neighborhood, the first tabu list has to be checked. By doing so, we avoid cycling, but we do allow an activity to be consecutively selected if the considered moves have the same direction. The aspiration criterion defines that a move that would yield a new best solution will be accepted even if it would normally be prohibited by the tabu list. Because the tabu search described here only adds or subtracts one unit of time buffer at a time, large shifts of the starting time of an activity compared to its initial starting time will only occur if all intermediate positions yield acceptable solutions. This might obstruct the procedure to move an activity into its actual best positioning for all other activity starting times considered fixed. To remove this inconvenience, one iteration step of the improvement heuristic of Section 5.1.4 will be allowed after a fixed number of iterations (100). The overall best found solution is stored throughout the whole procedure. The tabu search stops after a fixed number of iterations.

5.1.6 Exact algorithm

In this section, we propose a depth-first branch-and-bound procedure for inserting time buffers in the initial schedule. The procedure examines all possible buffer size combinations and tries to prune the search space by calculating bounds. It should be stressed that the pruning rules proposed in this section, are only valid if the resource flow network is preserved during project execution. We run the STC heuristic to obtain an initial upper bound UB on the stability cost.

The algorithm starts with a *preprocessing phase*. The activities are listed in decreasing order of their earliest starting time $es_j = s_j^U$ in the unbuffered initial baseline schedule (ties broken by decreasing activity number). For the input schedule of Figure 2.2 this would yield the list $L = (l_{[1]}, l_{[2]}, ..., l_{[n]}) = (9, 8, 6, 7, 5, 4, 3, 2, 1)$. We identify N_{j-} as the set of activities that precede j in the list and the set N_{j+} as the set of successors of activity j in the list.

For each activity j in the network $G(N, A \cup R)$, the latest start time ls_j is calculated given that $ls_n = \delta_n$. The *total float* TF_j of activity j equals $ls_j - es_j$. It denotes the maximum amount of time by which activity j may be delayed without violating the due date δ_n. The preprocessing phase works as described in Algorithm 5.3.

Algorithm 5.3 The preprocessing phase

$\forall j \in N$:

for $FF_j = 0$ **to** TF_j **do**

Simulate 100 executions scenarios of $G(N_{j+}, A \cup R)$ with:

$\forall i \in N_{j+} : s_i = es_i$

$s_j = es_j + FF_j$

Calculate $minstab_j(FF_j) = w_j E|\mathbf{s}_j - s_j|$

Because all activities that succeed j in the list L are started at their earliest starting time, $minstab_j(FF_j)$ is a lower bound on the stability cost induced by activity j when activity j is preceded by a buffer of FF_j time units.

After this preprocessing step, the algorithm starts with a depth-first search. The activities are considered in the order dictated by the list $L = (l_{[1]}, l_{[2]}, ..., l_{[n]})$. We identify the activity under consideration as the *current activity*. The first activity $l_{[1]}$ on the list is the dummy end activity n. We generate the root node at level 0 of the search tree with $s_n^{'} = \delta_n$. We move to the second current activity $c = l_{[2]}$ on the list, and generate the left most node at level 1 of the search tree with $s_c^{'} = s_c$. We compute a lower bound on the stability cost induced by this current activity c using simulation as

$$LB1_c = \sum_{l \in N_{c-}} w_l E|\mathbf{s}_l - s_l| \tag{5.7}$$

If $LB1_c \geq UB$, then the node can be fathomed, as it is not necessary to evaluate the starting times $s_i > es_i$ for any $i \in N_{c+}$, given that s'_l has been fixed for each $l \in N_{c-}$. Also, it is not necessary to evaluate later starting times for c. Hence, we can *backtrack* to the previous activity in the list.

Backtracking is done by moving one level up in the search tree and by investigating the next position $s'_{(c')} + 1$ for the current activity c' explored

at that level. If there exists no subsequent starting time of c' such that $s_{c'} + d_{c'} \leq \max_{\forall m:(c',m)\in(A\cup R)} s'(m)$, then we continue backtracking until we meet an activity with feasible subsequent starting times. When backtracking reaches the root of the search tree the algorithm stops.

If $LB1_c < UB$, we can compute a tighter lower bound $LB2_c$. We start by calculating latest starting times ls'_i for $i \in N_{c+}$ given that $\forall l \in (N_{c-} \cup \{c\})$: $ls'_l = s'_l$. For any activity $i \in (N_{c+} \cup \{c\})$, the expression $w_i E|\mathbf{s}_i - s_i| \geq minstab_i(ls'_i - es_i)$ holds for any schedule that would be generated in lower branches of the search tree. Aggregation yields

$$\sum_{i\in(N_{c+}\cup\{c\})} w_i E|\mathbf{s}_i - s_i| \geq \sum_{i\in(N_{c+}\cup\{c\})} minstab_i(ls'_i - es_i) \tag{5.8}$$

We have shown above that $\sum_{i\in N_{c-}} w_i E|\mathbf{s}_i - s_i| \geq LB1$ holds in the current branch of the search tree. This leads to

$$LB2 = LB1 + \sum_{i\in(N_{c+}\cup\{c\})} minstab_i(ls'_i - es_i) \leq \sum_{i\in N} w_i.E|\mathbf{s}_i - s_i| \tag{5.9}$$

LB2 is thus a lower bound on the stability cost for all schedules with $s_j = s'_j$ and $\forall\, l \in N_{j-} : s_l = s'_l$. The computations made in the preprocessing stage make this bound rather easy to compute. If $LB2 \geq UB$, the current node can be fathomed and we can continue the search by evaluating the next integer start time for current activity c. If $LB2 < UB$, the current branch needs to be further explored by branching on the next activity in the ordered list. When all activities i with $es_i > 0$ have been branched on and $LB2 < UB$, a new best solution has been found with stability cost *LB2*.

Figure 5.8 illustrates the search logic on a small fictive problem. The current best solution found has a stability cost of 500. The node corresponding to starting current activity l' at time $s'_{l'}$ could not be fathomed by either *LB1* or *LB2*. We thus have to branch to the next level, where activity j is the current activity. Starting activity j at its earliest starting time es_j yields a $LB1_j < UB$. The stability impact on the activities $l \in N_{j-}$ is not large enough to prune the current branch. However, an earliest starting time typically allows little slack in front of the activities that remain to be scheduled in lower levels of the search tree, resulting in small values of FF_i for $i \in N_{j+}$

and thus large values of $minstab_i(FF_i)$ and $LB2$. $LB2_j = 550$ allows us to fathom the node and to continue the search by evaluating $s_j = es_j + 1$. $LB1_j$ goes up to 400, but $LB2_j$ has decreased due to increased slack for the activities in N_{j+}. Because both lower bounds are smaller than the current best solution, we branch to node 4 at the next level of the search tree. Updating $LB1$ at the first branch of this level results in $LB1_{i'} \geq UB$. This branch and the subsequent branches can be pruned and we continue the search by backtracking to level j. The next starting time position of j, namely $es_j + 2$, also yields $LB1_j > UB$. We backtrack to the previous level. The procedure continues in this way until we are able to backtrack until the parent node.

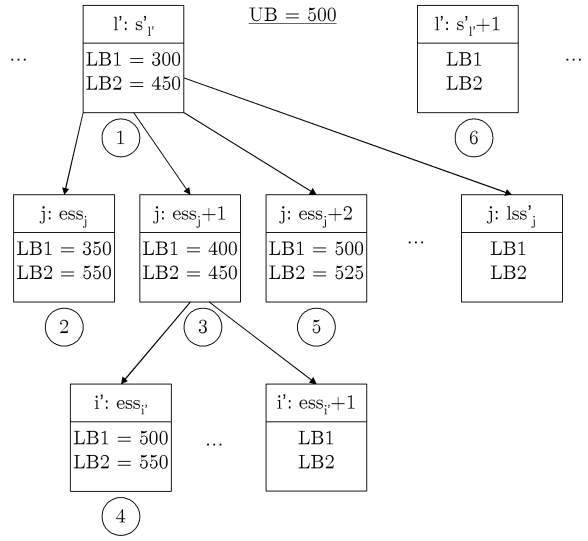


Figure 5.8: A partial branch-and-bound tree

Running the optimal buffer insertion algorithm on the schedule of Figure 2.2 with $\delta_n = 20$, yields the robust project schedule of Figure 5.9. Remark that although this schedule is very different from the STC schedule, its stability cost is only incrementally better in the simulation. Schedule 5.9 is better at protecting the project due date at the cost of a higher intermediate stability cost.

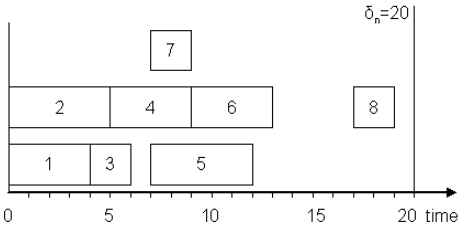


Figure 5.9: A robust buffered schedule

5.2 Experimental results

In this section, experimental results will be given for the different buffer insertion procedures proposed in this chapter. The results of the heuristic procedures are based on the results obtained in the paper of Van de Vonder et al. (2005a). Section 5.2.1 will shortly revise the main conclusions drawn in that paper for the simple heuristics, while Section 5.2.2 will do the same for the improvement procedures. We refer to the extensive experimental analysis of Chapter 8 for a more thorough examination of the most promising among the heuristics. Finally, results for the exact algorithm of Section 5.1.6 are discussed in Section 5.2.3.

5.2.1 Simple heuristics

All heuristic algorithms evaluated in this section and the following section have been coded in Microsoft Visual C++ 6.0. The procedures were tested on the well-known J30, J60 and J120 PSPLIB data sets (Kolisch & Sprecher 1997). For details on these instances we refer to the parameter settings section of the PSPLIB website (http://129.187.106.231/psplib/).

The heuristic RCPSP solution obtained by the combined crossover algorithm of Debels & Vanhoucke (2006) serves as the benchmark and is used as initial unbuffered schedule for all buffer insertion procedures. The project due date δ_n is set equal to $\lfloor 1.3 \times C_{\max} \rfloor$ which has been found to be an adequate due date for most project schedules in a recent study (Van de Vonder et al. 2006b). The random procedure of Artigues et al. (2003) resolves the resource allocation. Repairing the schedule on schedule breakage is done by

applying a robust serial schedule generation scheme (see Section 7.2) to the priority list that orders activities in non-decreasing order of their starting time in the baseline schedule. Ties are broken by decreasing order of activity weight, then by increasing activity number.

For every algorithm we calculate the average stability cost over all networks and 100 simulated execution scenarios on both a training set and a test set of simulated disruptions. The test set of executions is run to detect *overfitting* as will be explained in the next section. Besides the average, we also examine the percentage of network instances for which a certain algorithm yields the minimum stability cost among the algorithms within its class. Again, this measure will be calculated for training and test set disruptions to examine the degree of overfitting. We feel that comparing simple heuristics with improvement algorithms would give few additional insights. Also the average computational times (in seconds) are looked at for each algorithm.

The results obtained in Van de Vonder et al. (2005a) reveal that STC gives by far the best results among the simple heuristics. Although VADE and RFDFF have a much smaller stability cost than the unbuffered schedule, they only outperform STC on a few network instances. RFDFF encounters problems when dealing with networks with tight resource constraints. In these networks many resource conflicts need to be resolved, which will lead to more extra precedence relations in the resource flow network and eventually to a larger $C_{\max}$ for the RCPSP and thus a larger δ_n. RFDFF typically allocates a larger portion of the total safety to the dummy end activity than the other heuristics. For networks with tight resource constraints, the total safety included (recall that $\delta_n = \lfloor 1.3 \times C_{\max} \rfloor$) might be too high such that RFDFF overprotects the project completion and has to pay a high extra stability cost for this unnecessary extra protection due to poorly buffered intermediate activities. Also larger networks, i.e. networks with more activities, will in general result in a larger makespan and this explains the fact that RFDFF performs exceptionally bad on the J120 data set. RFDFF also scores relatively worse when the degree of uncertainty decreases because it typically overprotects the project completion, causing poorly buffered intermediate activities and unnecessary stability losses during execution.

The required computational effort is highly correlated with the number of time-consuming[2] evaluations needed in the procedure, which is low for all simple heuristics. The STC heuristic, for instance, runs on average in slightly less than 2 seconds for the PSPLIB J120 network instances. Because VADE selects the best out of a range of solutions, its computational time is the most demanding among the simple heuristics. Especially for the J120 networks the computational time of VADE exceeds this of the other simple procedures. For the same reason, VADE is slightly affected by overfitting. However, this overfitting is clearly not critical.

5.2.2 Improvement heuristics

The improvement heuristic of Section 5.1.4 can start from any feasible schedule. Results on the training set reveal that all stable project schedules generated by a simple buffer insertion heuristic (STC, VADE and RFDFF) provide good starting solutions for the improvement heuristic. Only starting from the unbuffered schedule leads to substantially higher stability cost and computational effort. As somewhat expected, the tabu search procedure of Section 5.1.5 already obtains the best results of all heuristics for a restricted number of iteration steps (100). However, the $2\times(n-2)$ evaluations required at each of the iteration steps of the tabu search procedure account for an increase in computation time. When $n = 120$, the required computational effort of the improvement heuristic becomes more demanding. Surprisingly, VADE results in the best starting solution for the descent approach in more cases than STC, while the latter has been shown to be the best simple heuristic.

The algorithms proposed in Sections 5.1.4 and 5.1.5 select the best neighborhood solution by evaluating several scenarios of disruptions drawn from the stochastic activity duration density functions. Including information about these simulated disruptions might make the buffer allocation decision process subject to overfitting to these disruption scenarios. The best schedule in the simulation experiment will not necessarily result in the best schedule for the disruptions that will occur during the actual project execu-

[2]Remember that each evaluation consists of 100 simulated executions of the project that have to be scheduled by using the robust parallel SGS.

tion, even if we assume that they are drawn from the same density functions. Hence, we also evaluate the schedules generated by the improvement heuristics on the test set of disruptions. More information about overfitting will be given in Section 8.3.3.

On the test set, we observe an overall increase in the average stability costs compared to the training set results, while the mutual differences in stability cost between the algorithms are largely reduced for J30 and J60. These small differences make it harder to justify the high computational burden of the tabu search procedure. We may conclude that especially the tabu search is subject to overfitting. When the degree of uncertainty decreases, the simple STC heuristic even obtains results that are very similar to the improvement heuristics. The local searches only overfit the baseline schedule on the simulated disruptions in the training set. Tabu search is certainly not recommended in a low variable environment.

We attempted to reduce the computational time of tabu search and the improvement phase, by replacing the objective function to evaluate the neighboring solutions by the surrogate objective of Eq. 5.6 which does not rely on simulation to be evaluated. However, it was found that it became very difficult to obtain satisfying results if the ultimate objective function remained to minimize $\sum_j w_j \left| s_j^0 - s_j^T \right|$.

5.2.3 Exact algorithm

The exact algorithm of Section 5.1.6 has not been included in the experiment of Van de Vonder et al. (2005a) because the required computational time is too extensive for the PSPLIB network instances. The practical use of the procedure is thus very limited.

Also, whether a solution is optimal depends on the reactive procedure used to evaluate the solution schedule. The exact algorithm of this chapter requires that the resource flows are kept fixed during execution (see Section 7.4.2) to guarantee optimality. If not, the lower bounds of Equations 5.7 and 5.9 are not applicable and without these bounds, the procedure would become a full enumeration of all possible solutions, which has been shown to be unrealistic earlier in this chapter. The procedure of Artigues et al. (2003) is again used to construct a feasible resource flow network.

The exact algorithm was tested on a problem test set that was constructed using the RanGen project scheduling network instances generator developed by Demeulemeester et al. (2003). All instances consist of 15 non-dummy activities. For all three project characteristics used in the experimental design to set op the data set (OS, RF, RC), two values are used, namely 0.3 and 0.7. The number of resource types in a project is fixed to four. For each of the 2^3 parameter settings, 10 network instances were generated, yielding a total of 80 test instances. The unbuffered baseline schedule is generated by the branch-and-bound code of Demeulemeester & Herroelen (1992, 1997) with a project due due δ_n that is set equal to a 30% prolongation of the project makespan. Remark that the required computational time will be highly correlated with δ_n. Activity duration disruptions and activity weights were drawn from the distribution that will be defined in the experimental set-up of Chapter 8.

Table 5.4 gives the obtained results on our eighty 15-activity networks. We compare them with the benchmark results obtained by the STC heuristic and by running the descent algorithm of Section 5.1.4 on the STC schedule (STC D) on both average solution robustness (in terms of the objective function of Eq. 2.3) and average required computational time on a Pentium IV 2.9 GHz workstation. We limited the computational time per network instance to 30 minutes if necessary. It should be remarked that 67 out of 80 network instances could be solved optimally well before this time limit. The average improvement of the exact algorithm is small compared to the improvement heuristic, while the computational time explodes. The exact algorithm ran on average just over 7 minutes, but the true average computational time will be higher due to the 30 minutes limit. The median computational time is much lower, i.e. 51 seconds.

The effectiveness of the lower bound mechanisms can be calculated by dividing the number of nodes visited by the total number of nodes in the branch-and-bound tree. This last number already becomes very large for most 15-activity networks. We stopped the counting procedure at 10^{10} nodes and noticed that for 68 out of the 80 networks in our data set, the search tree contains more nodes than that threshold. Our exact algorithm visits on average just over 10^6 nodes. The lower bound mechanisms could

Table 5.4: Results of the exact algorithm

	Stability	Time (in s.)
STC	21.22	$< 10^{-2}$
STC D	19.00	0.02
Exact	18.89	427.51

thus already prune well over 99% of the counted nodes in the search tree. This percentage would only increase by more precise calculations of the exact number of nodes in the search tree. For more complex networks, the number of nodes is exponentially large such that even after effective pruning a high computational burden persists.

5.3 Conclusions of this chapter

Proactive project scheduling is concerned with building stable baseline schedules that are able to absorb most of the anticipated disruptions during project execution. In this chapter, various heuristic algorithms and an exact algorithm for inserting time buffers in a project schedule were presented and evaluated.

STC was shown to be a relatively fast heuristic that performs well on small and large projects. It will be implemented in a real-life environment in Chapter 9. Improvement algorithms and the exact algorithm require simulation to evaluate several candidate solutions and become computationally infeasible for large-scaled projects. They may also be subject to overfitting. Trying to avoid these drawbacks of simulation-based approaches by working with surrogate objective measures leads to inferior results on solution robustness.

In this chapter buffers were allocated to an initial schedule S^U after resource allocation was decided by the algorithm of Artigues et al. (2003). The evaluation of these buffered schedules was mostly done by a priority rule based reactive procedure. The study of the impact of different initial schedules on the efficiency and effectiveness of the buffer allocation process

will be the main topic of the next chapter. In Chapter 8, a large-scale experiment investigates the interactions between initial schedule selection, resource allocation, buffer allocation and reactive procedures. Detailed results for some of the most promising buffer insertion procedures proposed in this chapter will be included in that experiment.

Chapter 6

Integrated solution robust project scheduling

So far our methodology for generating proactive project schedules relied on a two-stage approach. First, an initial baseline schedule was constructed by solving the RCPSP under the minimum makespan objective without looking at robustness. This schedule was then made solution robust in a second stage. In the previous chapters we looked at resource allocation (Chapter 4) and buffer insertion (Chapter 5) to improve the solution robustness of a baseline schedule.

The aim of this chapter is twofold. First, in the two-stage process to construct a stable project schedule one traditionally devotes little attention to the first stage, in which a feasible solution for the RCPSP is found. Second, although we emphasized the advantages of a two-stage approach, the interrelations between initial schedule selection, resource allocation and buffer insertion should not be ignored. Hence, robust scheduling approaches will be introduced that integrate the two-stage approach in one single stage.

The remainder of this chapter is organized as follows. In Section 6.1 a branch-and-bound procedure is proposed that finds all feasible active schedules with a makespan smaller than a given threshold. Sections 6.2 and 6.3 introduce two metaheuristics for initial schedule selection. The first one basically consists of applying the STC heuristic of Section 5.1.3 to a range of initial schedules. The second one relaxes the assumption that robust

scheduling should follow a two-stage approach by integrating initial schedule selection and buffer insertion. Experimental results with the proposed procedures are given in Section 6.4. Potential future research directions (Section 6.5) and some overall conclusions (Section 6.6) are presented at the end of the chapter.

6.1 A branch-and-bound algorithm for initial schedule selection

In previous chapters, a single initial schedule S^U is generated by applying one of the procedures from the extensive RCPSP literature. However, for most projects many minimum duration and near-minimum duration schedules may be selected depending on the searching mechanism of the applied procedure. These candidate initial schedules may differ in robustness. For our example of Table 2.1 two alternative optimal schedules exists in addition to the schedule of Figure 2.2. One of them has already been introduced in Figure 4.1.

We adapt the branch-and-bound procedure of Demeulemeester et al. (1992, 1997) so that it generates a family of optimal and near-optimal schedules instead of a single optimal solution. First, the procedure is run to solve the RCPSP instance to optimality in order to know the minimum makespan $\mathbf{C}^*_{\max}$. Then, we rerun the procedure in such a way that it finds *all* minimum duration schedules. For this purpose, several changes had to be made to the algorithm. First, the upper bound with the current best solution has to remain equal to $\mathbf{C}^*_{\max} + 1$ throughout the entire search. Second, all minimum duration schedules have to be stored. Last, despite their efficiency for solving the standard RCPSP, the dominance rules (f.i. the cutset dominance rule) needs to be suppressed in order to find all optimal solutions in the search space. Remark that this search space only comprises active schedules.

Using a minimum duration schedule as input schedule in the two-stage proactive scheduling procedure does not necessarily yield the most robust baseline schedule for a project with a predefined project due date δ_n. One of the objectives of the computational experiment in Section 6.4.1 is to inves-

tigate the impact on solution robustness of using near-minimum makespan schedules as initial schedules in step one of the two-stage procedure. In order to find a set of such near-optimal schedules for a given project instance, the above described procedure will be rerun several times with upper bound set equal to $\mathbf{C}^*_{\max} + \Delta + 1$, for $\Delta = 0, 1, 2 \ldots$. This allows us to find all schedules in the search space with a makespan equal to $\mathbf{C}^*_{\max} + \Delta$.

6.2 Metaheuristic 1

The RCPSP is known to be *NP*-hard in the strong sense (for proofs see Blazewicz et al. (1983) and Demeulemeester & Herroelen (2002)). Looking for multiple solutions as described above may very well be computationally inefficient.

In this section we create a metaheuristic that applies STC (see Section 5.1.3) to different feasible RCPSP solutions, trying to find the best combination of initial schedule and buffer insertion. Remember that resource allocation decisions need to be made before the STC algorithm can be applied. In this metaheuristic, we generate feasible resource flows using the procedure of Artigues et al. (2003). Compared to the branch-and-bound procedure of the previous section, this metaheuristic tries to guide the search to a solution that performs well on the stability objective function of Eq. 2.3. The main drawback of the approach followed by Metaheuristic 1 is that the STC buffer insertion procedure needs to be applied numerous times. Hence, Metaheuristic 1 can become rather demanding on computation time for large projects, because the STC heuristic needs almost two seconds for a 120-activity network.

6.2.1 The algorithm

In this section, the implementation details of the first metaheuristic are described. Two sets of schedules are maintained throughout the search. *EliteSet* contains solutions for the solution robustness maximization problem that will hopefully improve through the iterations. Schedules will be buffered by applying STC to an initial schedule S^U before their inclusion in *EliteSet*. They will be stored by maintaining an activity list and a buffer vector (see

below). *RCPSPSet* is formed with (unbuffered) solutions for the RCPSP that can be stored by only maintaining an activity list. The schedules in *RCPSPSet* help to create new initial schedule candidates. We will denote the cardinality of *EliteSet* and *RCPSPSet* respectively as *nelite* and *nRCPSP*. Typically, *nelite* will be substantially smaller than *nRCPSP*. The parameter settings for these and other parameters in our experimental analysis will be given in Section 6.4.2.

Algorithm 6.1 Metaheuristic 1

1. Initialize *EliteSet* and *RCPSPSet*
2. For $i = 0$ to (*niter* -1)
   (a) (*Candidate*, *CandRCPSP*) = Path-Relinking(*EliteSet*,*RCPSPSet*)
   (b) Evaluate *Candidate*
   (c) Update(*EliteSet*, *RCPSPSet*, *Candidate*, *CandRCPSP*)

The procedure is summarized in Algorithm 6.1. The subsequent steps in this algorithm are explained in more detail in the upcoming sections. After initialization (see Section 6.2.1.1), new candidates for the *EliteSet* and *RCPSPSet* are calculated with a Path-Relinking procedure at each iteration step. These candidates are evaluated (see Section 6.2.1.3) and replace solutions in the sets according to some criteria as will be explained in Section 6.2.1.4. The algorithm terminates when a predefined number of iterations (*niter*) has been reached.

6.2.1.1 Initialization

We create *nelite*+ *nRCPSP* schedules with the regret-based biased random sampling procedure of Drexl (1991) under the latest finishing time rule. We apply the *double justification DJ* (see Valls et al. (2005)) to all of them. Double justification $DJ(S)$ of a schedule S applies a left justification to the right justification of S. *Left (right) justification* advances (delays) each

activity of S in such a way that it cannot start earlier (finish later) without delaying (advancing) some other activity, or violating the constraints, or increasing the makespan. It is a proven technique to improve solutions for the RCPSP without requiring much computational time. This initial solution strategy is used in *HGA* (cf. Valls et al. (2002)), one of the best heuristic algorithms for the RCPSP, to create the initial population. Once the schedules have been created, we order the solutions in increasing order of their makespan. The *nelite* best solutions on makespan are evaluated for stability (see Section 6.2.1.3) and form the *EliteSet*. The remaining *nRCPSP* solutions form the initial *RCPSPSet*.

6.2.1.2 Path relinking

Algorithm 6.2 describes the path-relinking procedure that heavily relies on the procedure PR of Algorithm 6.3. *EliteSet*(i) (*RCPSPSet*(i)) denotes the i-th solution in the set. *Candidate*(i) represents a solution, but *CandRCPSP* is a set of solutions. The procedure PR requires two schedules A and B and a parameter *nsol* as input and returns a set of solutions or the empty set $\emptyset$. It will be described in detail in Algorithm 6.3.

In the procedure PR we denote a schedule by *topological ordering* (TO)(cf. Valls et al. (1999)) instead of a traditional priority list. Given a schedule S, any vector $\gamma = (\gamma_1, \ldots, \gamma_n)$ of different integer numbers between 1 and n that satisfies: $s_i < s_j \rightarrow \gamma_i < \gamma_j$ is called a TO representation of schedule S. We can switch between a TO and an activity list λ with the equation $\gamma_i = \lambda^{-1}(i)$ if λ satisfies $s_i < s_j \rightarrow \lambda^{-1}(i) < \lambda^{-1}(j)$. Throughout the algorithms we work with this type of activity lists. For an example priority list $\lambda = (3, 1, 2)$ we see that activity 3 occupies the first position (i.e. $\lambda^{-1}(3) = 1$) such that $\gamma_3 = 1$. The TO representation conform with λ is $\gamma = (2, 3, 1)$.

In Step 6 (e) of Algorithm 6.3 the serial SGS is applied to generate S by selecting at each iteration the eligible activity with smallest weight γ_j^3. A similar operator was used in Valls et al. (2005).

Algorithm 6.2 Path relinking

For $i = 0$ to $(nelite - 1)$ do
  {
  For $j = 0$ to $(nelite - 1)$ do
    {
    $S =$ Solution with smallest makespan calculated in the procedure
      $PR(EliteSet(i), EliteSet(j), nsol1)$
    If ($Candidate(i) = \emptyset$ or S has smaller makespan than $Candidate(i)$)
      $Candidate(i) = S$
    }
  For $j = 0$ to $(nRCPSP - 1)$ do
    {
    $S =$ Solution with smallest makespan calculated in the procedure
      $PR(EliteSet(i), RCPSPSet(j), nsol2)$
    If ($Candidate(i) = \emptyset$ or S has smaller makespan than $Candidate(i)$)
      $Candidate(i) = S$
    }
  }
For $i = 0$ to $(nRCPSP - 1)$ do
  {
  $j = 0$ to $(nRCPSP - 1)$ do
    {
    $CandRCPSP =$
      $CandRCPSP \cup PR(RCPSPSet(i), RCPSPSet(j), nsol2)$
    }
  }

Algorithm 6.3 PR(A, B, $nsol$)

1. Deduce activity lists λ_1 and λ_2 from schedules A and B
2. Set $= \emptyset$
3. Transform λ_1 and λ_2 in TO's γ^1 and γ^2
4. Let $N' = N$
5. Let $nact = |N'|/(nsol + 1)$
6. For $i = 0$ to $(nsol - 1)$ do
   (a) Randomly select $nact$ activities from N'
   (b) Change = Change $\cup$ {selected activities}
   (c) $N' = N' \setminus$ {selected activities}
   (d) $\forall j \in N$:
      i. If $j \in$ Change: $\gamma_j^3 = \gamma_j^2$
      ii. Else: $\gamma_j^3 = \gamma_j^1$
   (e) Create a schedule S with the weights γ_j^3
   (f) $S = DJ(S)$
   (g) If S is different from A, B and the solutions in Set: Set $\cup$ S
7. Return Set

6.2.1.3 Evaluate

We evaluate *Candidate*(i) for $i = 1, \ldots,$ *nelite* as follows. We first generate an initial unbuffered schedule S^U by applying a serial SGS to the activity list λ pertaining to *Candidate*(i) and then add buffers by applying the STC heuristic to this initial schedule. Afterwards the objective function of Eq. 2.3 is evaluated through simulation on a training set with 100 repetitions. If there are more than *nelite* schedules in *Candidate*, they are not evaluated. This means that we evaluate at most *nelite* solutions at each iteration step. Sometimes fewer solutions are evaluated because *Candidate*(i) might be empty as can be seen in Step 6 (g) of Algorithm 6.3.

6.2.1.4 Update

Updating the sets is done as follows. *Candidate*(i) replaces *EliteSet*(i) when it scores better on the objective function and the best of the list *CandRCPSP* replaces *RCPSPSet*(i) when it performs better on makespan.

6.3 Metaheuristic 2

The first metaheuristic has two main drawbacks. First of all, we cannot evaluate many solutions, since STC becomes CPU critical for multiple procedure calls for large networks. Secondly, not all solutions for our problem can be reached through the combination of an active initial schedule and the application of STC to this schedule to insert buffers. Possible buffer insertions that are not generated by STC will be unjustly excluded from consideration. In order to solve these problems we have created the second metaheuristic that incorporates the buffers in the codification. This means that next to an activity list λ and an SGS, a solution is also represented by a vector of buffers β. For a given representation, a schedule is built by applying the serial SGS to the activity list and then adding the buffers that were stored in the buffer vector. Adding a buffer to activity i means that the activity itself and all its successors in $T(A \cup R)$ are delayed by $\beta(i)$, the i-th element in the buffer vector. A buffer vector $\beta = (0, 3, 0)$ for example indicates that activity 2 and its successors will be delayed by three time units compared to the schedule generated by applying the serial SGS to λ.

Remark that as in the first metaheuristic, buffer insertion only occurs after resource flow network $G(N, R)$ has been fixed.

The buffers stored in β are usually far from optimal and worse than those provided by STC. However, they evolve with the solutions, becoming better through evolution. The idea is that it is a waste of time to calculate (near) optimal buffers (by STC) for the solutions in the early stages of the algorithm, because they are going to be replaced by better solutions later in the search. Buffer insertion and initial schedule selection have been integrated.

Recently, a similar approach has been proposed by Ballestín & Leus (2006) in single-machine scheduling. The tabu search for buffer insertion introduced in Section 5.1.5 and the tabu search of Lambrechts et al. (2006b) for robust scheduling under stochastic resource availabilities also rely on similar buffer list representations.

6.3.1 The algorithm

The proposed metaheuristic is a population-based algorithm. A population of solutions (POP) is created and maintained through several iterations. There is also an auxiliary population *PopRCPSP* formed by RCPSP solutions. This population is used to refresh the main population. POP is initialized by creating *ninitial* schedules with the regret-based biased random sampling procedure under the LFT rule (see also Section 6.2.1.1). We apply the above described double justification algorithm DJ to all of them. We heuristically insert buffers in these solutions by adding iteratively one extra buffer size unit to *ininact* activities j selected from N with probabilities proportional to $w_j \sum_{\forall i:(i,j)\in T(A\cup R)} P(\mathbf{d}_i > d_i + MSPF_{i,j})$ (cf. Section 5.1.3). This iterative procedure terminates when a predefined number (= *ininiter*) of iterations without improvement on the surrogate objective function of Eq. 5.6 are reached. Finally, we select the best *nPop* solutions according to their performance on the surrogate objective function and evaluate them by simulation. For population *PopRCPSP*, we create *nPopRCPSP* schedules with the regret-based biased random sampling procedure under the LFT rule and apply the double justification algorithm to all of them.

At each iteration one of four possible operators is chosen and applied to one or two solutions of the population. The probabilities of selecting operators A, B, C and D are 1/8, 3/8, 1/4 and 1/4 respectively. When operator B or special operator D is selected, the auxiliary population will be modified. When we introduce a solution in *POP*, we eliminate the worst solution in it. When we select a solution of *POP* (*PopRCPSP*), those with a better makespan have a bigger probability of being selected. The second metaheuristic is summarized in Algorithm 6.4. More details on the operators are given in the next section.

Algorithm 6.4 Metaheuristic 2

1. Calculate initial population *POP* and auxiliary population *PopRCPSP*
2. While *termination criterion not met*
   (a) Randomly select operator X from {A, B, C, D}
   (b) i. If X is binary operator A then select two solutions of *POP* and apply operator A to obtain two solutions that are included in *POP* if they are better than any in *POP*
   ii. If X is binary operator B then select one solution of *POP* and one of *PopRCPSP* (*S*). Apply operator B and obtain one solution that is included in *POP* if it is better than any in *POP*. Obtain one solution that replaces *S* in *PopRCPSP*
   iii. If X is unary operator C then select one solution of *POP*. Change the buffers and evaluate it. It is introduced in *POP* if it is better than any in *POP*
   iv. If X is special operator D: apply several iterations of the DJGA algorithm to the auxiliary population *PopRCPSP*.

6.3.2 Operators

In this section, we describe the operators that are applied in Algorithm 6.4.

6.3.2.1 Operator A

Operator A returns a pair of new activity lists and is a generalization of the two-point crossover used in the genetic algorithm of Hartmann (1998). We draw two random integers $q1$ and $q2$ with $1 \leq q1 < q2 \leq n$. λ_M and λ_F are the activity lists corresponding to the mother solution and the father solution respectively. β_M, β_F and β_D denote the buffer vectors. The daughter's activity list λ_D is determined as follows:

- $\lambda_D(p) = \lambda_M(p); \quad \beta_D(\lambda_D(p)) = \beta_M(\lambda_D(p))$; $\quad p = 1, \ldots, q1;$
- $\lambda_D(p) = arg\min_{k \in N}\{\lambda_F^{-1}(p) | k \notin \{\lambda_D(1), \ldots, \lambda_D(p-1)\}\};$
  $\beta_D(\lambda_D(p)) = \beta_F(\lambda_D(p)); \quad p = q1+1, \ldots, q2;$
- $\lambda_D(p) = arg\min_{k \in N}\{\lambda_M^{-1}(p) | k \notin \{\lambda_D(1), \ldots, \lambda_D(p-1)\}\};$
  $\beta_D(\lambda_D(p)) = \beta_M(\lambda_D(p)); \quad p = q2+1, \ldots, n;$

A simple example will clarify this operator. Assume a 5-activity network, with the mother solution represented by $\lambda_M = (1, 2, 3, 4, 5)$ and $\beta_M = (0, 4, 3, 0, 2)$ and the father represented by $\lambda_F = (5, 1, 3, 2, 4)$ and $\beta_M = (0, 4, 0, 1, 0)$. $q1$ and $q2$ equal 1 and 3 respectively. The first $q1 = 1$ activities of the daughter are copied from the mother resulting in $\lambda_D = (1, x, x, x, x)$ and $\beta_D = (0, x, x, x, x)$, in which x denotes that the corresponding element is yet undecided. Afterwards, the daughter proceeds with the first $q2-q1$ activities from the father that are not yet in λ_D. This means that activities 5 and 3 are added from the father resulting in $\lambda_D = (1, 5, 3, x, x)$ and $\beta_D = (0, x, 0, x, 0)$. The daughter is completed by adding the remaining activities in the order of their appearance in λ_M. The representation of the daughter is: $\lambda_D = (1, 5, 3, 2, 4)$ and $\beta_D = (0, 4, 0, 0, 0)$.

Afterwards we apply a mutation, both to λ_D and β_D. For the activity list we have borrowed the mutation from Hartmann (1998) that states that two subsequent activities in the activity list are swapped with a certain

probability if the resulting sequence remains precedence feasible. To mutate β_D we randomly choose some activities and increase or decrease their buffers. A sibling is created symmetrically, by swapping the roles of the mother and father solutions.

6.3.2.2 Operator B

Operator B is similar to Operator A. The only difference is that one of the solutions is taken from the *PopRCPSP* and consequently has the null vector as buffer vector. The operator returns two solutions, described by an activity list and a buffer list. The best one in makespan is added to *POP* after buffer insertion and evaluation, the other one is added to *PopRCPSP* with the null vector as buffer vector.

6.3.2.3 Operator C

Operator C is in fact a family of operators, since there are three of them, all trying to improve the buffers of a solution. When operator C is selected, one of three family members is picked randomly. Selecting activities in operators C1 and C2 is not done randomly. Activities with high stability cost and small current buffer sizes are more (less) likely to be selected when we select activities in order to increase (decrease) buffer sizes.

Operator C1 starts from the input schedule S that can be represented by activity list λ_S and buffer vector β_S. It increases the buffers of *nactC1* selected activities and reduces the buffers of another *nactC1*. This results in schedule S^* with $\lambda^*_S = \lambda_S$ and with the updated buffer vector β^*_S. Then it calculates the value on the surrogate objective function of Eq. 5.6 for S^*. If this value is less than the objective value of schedule S, we repeat the procedure with $S = S^*$. Else we return to the original schedule S and choose new activities. This is done iteratively until a certain number (*niterC1*) of iterations without improvement is reached.

Operator C2 randomly chooses *nactC2* activities with repetition allowed and decreases their buffers by 1 time unit. Then it applies the STC procedure of Section 5.1.3, but it is stopped after a certain number of iterations *niterC2*.

Operator C3 is the complete STC procedure. Note that applying STC needs less time than in the first metaheuristic, since we begin the procedure with some buffers already included in the schedule.

6.3.2.4 Operator D

Operator D is applied to the auxiliary set of unbuffered schedules. It applies the DJGA procedure (Valls et al. 2005), which is a simple but very effective metaheuristic for the RCPSP. An iteration of this algorithm runs as follows. The population *PopRCPSP* is divided in pairs and the two-point crossover is applied to the $nPopRCPSP/2$ pairs, obtaining *nPopRCPSP* new solutions, which are mutated, evaluated on makespan through applying the serial SGS and to which the double justification procedure is applied. The population for the next iteration is formed by selecting the *nPopRCPSP* solutions with smallest makespan from all solutions in the original and the newly generated population.

Remark that we will not apply the above when one of the following conditions hold:

- the makespan of the best schedule in *PopRCPSP* equals the makespan of the worst;
- a certain number (*niterD*) of iterations without improvement has passed.

If either condition holds, a mutation is applied to refresh the population.

6.4 Experimental results

In this section we discuss the experimental results that we obtained for the procedures described in this chapter. We will focus on both initial schedule selection and the integration between this selection and robustness decisions discussed in previous chapters. Section 6.4.1 will discuss the major experimental findings for the branch-and-bound procedure, while Section 6.4.2 is devoted to results of the metaheuristic procedures. The actual parameter settings for the multiple parameters in the metaheuristic procedures will also be discussed.

6.4.1 Results of the branch-and-bound algorithm

The branch-and-bound procedure of Section 6.1 has been tested on the 480 PSPLIB J30 network instances. The number of alternative minimum makespan schedules varies enormously over the network instances. Many projects have a unique active RCPSP solution. In the J30 set, 120 network instances have a resource strength (Cooper 1976) $RS = 1$. For these networks, the early start schedule will by definition equal the unique minimum duration schedule because the peak resource requirement in this early start schedule equals the resource availability a_k for each resource type k. Nine other networks with $RS \neq 1$ show to have a unique solution. On the other hand, the number of alternative minimum duration schedules easily exceeds 10,000 for 25 network instances.

We ran the branch-and-bound several times with $\Delta = 0, 1, 2, 3, \ldots, 9$ to investigate the impact of the initial schedule makespan on stability. The first iteration results in one minimum duration schedule with minimum makespan $\mathbf{C}^*_{\max}$. The due date δ_n will be set equal to $1.3 \times \mathbf{C}^*_{\max}$, rounded to the closest integer value. In each subsequent iteration, the best stability cost and average stability cost for the obtained schedules with makespan $\leq \mathbf{C}^*_{\max} + \Delta$ are calculated. If necessary, we truncate each run of the branch-and-bound code once a new solution has been found after more than one minute of computational time[1] or whenever 100 solutions with makespan equal to $\mathbf{C}^*_{\max} + \Delta$ have been found. The project executions were simulated by respecting the resource flows that were decided by the MABO procedure.

We observe in Figure 6.1 that predictive schedules with near-optimal makespan ($\Delta > 0$) may on average (dashed line) result in a modestly improved stability cost compared to minimum duration schedules. A schedule with a larger makespan typically incorporates more slack for the intermediate activities. This will have a positive impact on stability cost. However, an increasing makespan of the predictive schedule will also lead to an increased percentage of due date violations. For increasing Δ, this negative effect will become dominant and result in an overall negative effect on solution robustness. The full line in Figure 6.1 represents for each Δ the lowest stability cost among the schedules with makespan $\leq \mathbf{C}^*_{\max} + \Delta$. This curve

[1] Remark that this can take much more time than 1 minute.

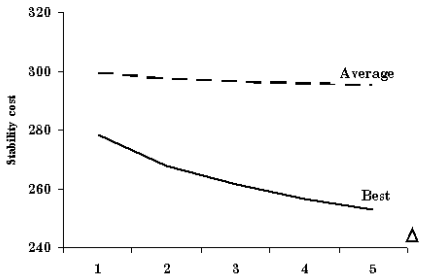


Figure 6.1: The impact of schedule makespan on stability

is by definition non-increasing.

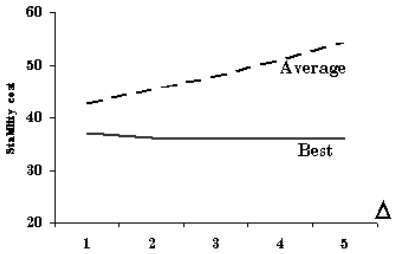


Figure 6.2: The impact of schedule makespan on stability after buffer insertion

When time buffers are inserted to make the predictive schedule more robust, the stability cost advantage of starting from a near-optimal initial schedule disappears. In Figure 6.2, the average expected stability costs for the STC schedules are represented by the dashed curve. The slack already present in the non-optimal unbuffered schedules improves robustness, but also reduces the freedom to add buffers (slack) by the buffer insertion procedure. STC is a very effective buffering procedure so that it pays off to launch this procedure on a minimum duration schedule. We remark that there is also hardly any improvement for the best schedule found when we allow to start from non-minimum duration schedules. This also shows that it is advantageous to allot more computational time to a single run of

the branch-and-bound procedure in which we search for minimum duration schedules only ($\Delta = 0$), rather than to consider near-optimal schedules in multiple runs. More detailed results of this approach will be given in the computational experiment of Chapter 8.

When we are interested in finding a solution robust baseline schedule, we need to draw attention to the integration of initial scheduling and buffer insertion. Clever initial schedule selection can substantially improve robustness, but we are unable to simply select the unbuffered schedule with the lowest simulated stability cost or the lowest lower bound for stability (see Section 4.2.3) and then add buffers to this schedule. It is recommended to evaluate candidate schedules after buffer insertion. A simple heuristic such as STC is required to allow for the evaluation of several candidate baseline schedules.

6.4.2 Results of the metaheuristic procedures

In an experimental set-up with parameter settings equal to these of the high variance, loose due date and high *wp* case that will be discussed in Section 8.2, we have compared the performance of Metaheuristics 1 and 2 to benchmark results obtained by applying the STC procedure to an initial schedule obtained by the combined crossover algorithm developed by Debels & Vanhoucke (2006). Results are shown in Table 6.1.

Parameter settings are important for metaheuristics to achieve a good performance. For PSPLIB J120, we set the parameters of Metaheuristic 1 as indicated in Table 6.2. Remark that *nelite* and *niter* are chosen such that the total number of STC calls is less than 50. Indeed, we need *nelite* STC applications in the initialization phase of Section 6.2.1.1 and maximally *nelite* applications for each of the *niter* times that the set *Candidate* is evaluated. The parameter settings for the second metaheuristic are summarized in Table 6.3.

Metaheuristic 1 obtains an improvement of more than 4% on the average stability cost for the 600 J120 network instances by applying STC to several (i.e. $niter \times nelite$) initial RCPSP solutions. The average computational time required is 96 s, which roughly corresponds to the average computational time required for 50 runs of the STC procedure.

Table 6.1: Results on PSPLIB J120

Procedure	Stability	Time (s)
STC	150,07	1.69
Metaheuristic 1	143.52	96.10
Metaheuristic 2	131.68	62.30

Table 6.2: Parameter settings for Metaheuristic 1

Parameter	Value
nelite	3
nRCPSP	30
niter	15
nsol1	9
nsol2	3

Table 6.3: Parameter settings for Metaheuristic 2

Parameter	Value
nPop	15
nPopRCPSP	500
ninitial	30
ininiter	10
ininact	5
nactC1	5
niterC1	10
nactC2	20
niterC2	60
niterD	5

The second metaheuristic is able to decrease the average stability cost with 12% in on average 62.3 s compared to the benchmark results. Better results are obtained for more than 500 out of 600 network instances. Integrated buffer insertion and schedule selection pays off.

It should be mentioned that, contrary to the branch-and-bound procedure and the first metaheuristic, this procedure does no longer rely on the STC procedure such that it is hard to tell whether its improvement is based on a better initial schedule selection or rather on an insertion of better buffers (stored in β) than those that would have been added by STC to the initial schedule. In order to exclusively investigate the improvement on the subject of initial schedule selection, the output schedule of Metaheuristic 2 should be either unbuffered or used as input for a more advanced buffer insertion procedure (see Chapter 5). This will be done in Chapter 8.

6.5 Future research

The main difference between the second metaheuristic and the two-stage approach of previous chapters is that the RCPSP optimization now occurs simultaneously with the optimization of the stability problem. We no longer look for a good RCPSP solution in stage one and buffer this solution in stage two. Metaheuristic 2 directly solves the stability problem in one integrated phase.

However, we are still in a way solving the RCPSP in both metaheuristics proposed in this chapter. This approach has as major advantage that it allows us to integrate very efficient procedures and techniques from the extensive RCPSP literature. The main drawback arises from the fact that, in contrast to the RCPSP (see Kolisch (1996b)), there does not necessarily exist an optimal initial schedule for the stability problem in the class of active schedules. All solutions obtained by the algorithms of this chapter add buffers to an active initial schedule S^U. During the buffer insertion process, the resource allocation decisions made on S^U will be respected. Consider a small 3-activity project with activity durations $d_1 = 3$, $d_2 = 2$ and $d_3 = 2$; with resource requirements $r_1 = 2$, $r_2 = 3$ and $r_3 = 2$ for a single renewable resource type with $a = 4$ and with a precedence relationship between

activities 1 and 2. The sole active initial schedule for this project will have activity starting times $s_1 = 0$, $s_2 = 3$ and $s_3 = 0$. Following the definition of an unavoidable resource arc in Eq. 4.1, we remark that any feasible resource allocation for this schedule contains a strictly positive resource flow between activities 3 and 2. These resource flows would remain intact for the buffered baseline schedules generated by the procedures of this chapter. The non-active schedule depicted in Figure 6.3 in which activity 2 precedes activity 3 will thus never be generated by any of them.

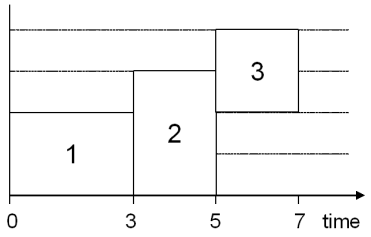


Figure 6.3: A non-active baseline schedule

Recently, we have been experimenting with a new metaheuristic in which buffered schedules are generated by applying a serial SGS to priority lists, but where the activity only becomes eligible for scheduling at its first feasible starting time augmented by a time buffer that is stored in a buffer vector β. In such an approach, the priority list $\lambda = (1, 3, 2)$ and the buffer vector $\beta = (0, 0, 4)$ would result in the schedule of Figure 6.3. Activity 3 has a first feasible starting time of 0 and a buffer $\beta(3) = 4$. It thus becomes eligible for scheduling at time 4 and will be scheduled at its first feasible starting time after 4, i.e. decision time 5.

This new metaheuristic would completely integrate initial schedule selection and buffer insertion and would make the resource allocation problem redundant (except when resource flows are fixed as a reactive procedure). It should, however, be mentioned that the enlarged solution space seriously affects the computational time required to obtain a satisfactory solution. Moreover, it may be intuitively unlikely that the extra solution space contains many solution robust schedules.

6.6 Conclusions

In the two-stage approach applied in this thesis we first solve the RCPSP to find an initial schedule and afterwards augment its solution robustness. The commonly overlooked impact of initial schedule selection on solution robustness has been investigated in this chapter. An exact branch-and-bound algorithm and two metaheuristics have been proposed to deal with this problem.

We found that drawing attention to the initial schedule selection pays off in terms of solution robustness, but in general requires a lot of computational time. For the branch-and-bound procedure we can somewhat reduce this computational burden by restricting the search to minimum duration schedules, without comprising much on stability. This approach will be taken in the computational experiment of Chapter 8.

The second metaheuristic relaxed the assumption that a two-stage approach for solution robust project scheduling should be applied. The initial scheduling problem and buffer insertion problem were solved in an integrated approach. The promising results of Metaheuristic 2 showed that initial schedule selection and buffer allocation are heavily interrelated and that such an integrated approach should be considered. Metaheuristic 2 will also be included in the large-scale computational experiment of Chapter 8.

Chapter 7

Reactive scheduling policies

In the previous chapters we discussed baseline scheduling methods with a different ability to absorb disruptions. It is to be expected that for a number of reasons none of them will ever be stable enough to cope with all possible disruptions that may occur during project execution:

- proactive scheduling only employs statistical knowledge of disruptions;
- the process of obtaining this knowledge is subject to estimation errors;
- anticipating all possible disruptions would simply be infeasible.

A proactive scheduling procedure must therefore be combined with a reactive scheduling procedure that, during schedule execution, allows to react to schedule disturbances that cannot be absorbed by the proactive schedule.

In Section 7.1, reactive scheduling is monitored as a multi-stage decision process. Before several robust reactive procedures are proposed in Section 7.4, robust schedule generation schemes and its properties are introduced in Sections 7.2 and 7.3. Some experimental results (Sections 7.5) and conclusions (Section 7.6) conclude the chapter.

7.1 Reactive scheduling as a multi-stage decision process

As stated above, we advise to react whenever the current project schedule becomes unfeasible. Although this approach might be considered un-

suitable in machine scheduling[1], the large-scale nature of projects urges for such a controlled execution. Essentially, proactive scheduling aims to reduce schedule nervousness by limiting the need for rescheduling decisions.

While the proactive scheduling efforts of the previous chapters are made before the actual start of the project, reactive project scheduling is a multi-stage decision process that takes place during project execution. To construct a predictive project schedule S^0, best practice is to employ a priori statistical knowledge about the stochastic project entities. Mostly, the expected or average duration $E(\mathbf{d}_j)$ of activity j is used to decide upon its predictive starting time s_j^0. The actual duration $\mathbf{d}_j$ of project activity j is only known with certainty at its completion. New information thus becomes gradually available during project execution, possibly requiring a schedule revision. At every point t in time, the *projected schedule* S^t (see Section 2.1.2.4) predicts how the project scheduler expects the project to unfold given the information available at that time.

If an activity was *projected* to have finished at time t, but it has not, the activity should remain active in the projected schedule. In Van de Vonder et al. (2005b), we explicitly assumed that in that case, the exact realized duration is known and the remainder of the disrupted activity could readily be scheduled for its remaining duration $\mathbf{d}_j - E(\mathbf{d}_j)$. This assumption might be unrealistic and will be dropped in this chapter. The disrupted activity will be continued for only one time period between t and $t+1$ and its continuation will be reconsidered iteratively until it finishes.

When the project finishes at stochastic time T, the projected schedule becomes the *realized schedule* S^T, that provides complete information about the actual realizations of the activities. In a stochastic environment, the realized schedule will be typically unknown before the project completion time T. We will refer to this stochastic schedule by the variable $\mathbf{S}^T$.

The desired reactive scheduling procedure to generate a projected schedule S^t at each decision time t depends on the objective function of the project. Often a distinction is made between *rescheduling* and *schedule repair*. Rescheduling means that the original scheduling problem is resolved

[1] In machine scheduling (see Vieira et al. (2003)) rescheduling occurs either periodically or event driven.

for the remainder of the project at schedule breakage. According to the robustness definitions of Chapter 2, this can be seen as quality robust reactive scheduling.

When solution robustness comes into play, such rescheduling would no longer be recommended. Solution robust reactive scheduling needs to ensure that the decisions made to construct a projected schedule S^t during project execution result in a small deviation $\Delta(S^0, S^T)$. The baseline schedule is thus repaired as well as possible at schedule breakage. The objective function used to evaluate this performance metric is the minimum expected stability cost function monitored in Eq. 2.3. We assume that the project is subject to a predefined project due date δ_n and that the baseline starting time of the dummy end activity s_n^0 is set equal to this δ_n. The cost of finishing the project before δ_n is zero, while the unit cost of surpassing the due date equals w_n.

The literature concerning solution robust reactive project scheduling is virtually void. Yu & Qi (2004) describe an ILP model for the multi-mode RCPSP and report on computational results obtained by a hybrid mixed integer programming/constraint propagation approach for minimizing the schedule deviation caused by a single disruption induced by a known increase in the duration of a single activity.

7.2 Schedule generation schemes

Reactive scheduling commonly relies on the application of so-called *scheduling policies* or *scheduling strategies* (Möhring et al. (1984, 1985)) under the objective of minimizing the expected makespan. A scheduling policy can be defined as a decision process that defines which set of activities are started at certain decision points t.

The best-known class of scheduling policies is the class of *priority policies* which order all activities according to a priority list λ and at every decision point select the next activities to start based on this priority list.

Often this selection occurs by applying a *parallel schedule generation scheme* (*parallel* SGS). The parallel SGS iterates over time and starts at each decision time t as many unscheduled activities as possible in accordance with

the precedence and resource constraints. The priority list dictates the order in which activities are considered.

In deterministic project scheduling, the *serial* SGS is the best-known alternative for the parallel SGS to decide which activities to start at what decision time. In each iteration, the next unscheduled activity in the priority list is selected and assigned the first possible starting time that satisfies the precedence and resource constraints.

To apply the serial SGS in a stochastic multi-stage decision process such as reactive scheduling, at any decision time a complete schedule has to be generated by applying a deterministic serial SGS. The observed past may not be altered and only activities that have an assigned earliest possible starting time that equals the current decision time are actually started. References to the serial SGS later in this paper, will always imply this *reactive serial SGS*, unless stated differently.

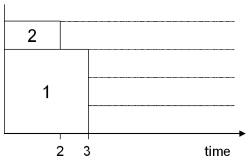


Figure 7.1: Partial schedule at time 2

The *stochastic serial* SGS (Ballestín 2006) used in an *activity-based priority policy* (Stork 2001) works as the deterministic serial SGS but adds the side constraint that $s_i \leq s_j$ for any activity i that precedes activity j in the priority list. The stochastic serial SGS might very well result in a different scheduling decision than the deterministic serial SGS. Consider for example the resource profile shown in Figure 7.1 for a project with a single renewable resource type. Assume that at decision time 2, activity 3 (with expected duration $E(\mathbf{d}_3) = 3$ and per period resource requirement $r_3 = 2$) and activity 4 ($E(\mathbf{d}_4) = 4$ and $r_4 = 1$) are eligible for scheduling. The stochastic serial SGS with priority list $\lambda = (1, 2, 3, 4)$ will not start activity 4 at time 2 because activity 3, a predecessor of 4 in the list λ, has not yet been scheduled. The deterministic serial SGS, on the other hand, will first project activity 3 at time 3 and will decide that this decision does not

impede the start of activity 4 at decision time 2.

Remember that we are primarily interested in robust reactive scheduling procedures, while the previously described SGSs are all concerned with minimizing the expected project makespan. Hence, we propose two new SGSs, namely the *robust parallel* SGS and the *robust serial* SGS.

The *robust parallel* SGS operates similarly to the parallel SGS with the side constraint that an activity j is only eligible to be scheduled if the current decision time $t \geq s_j^0$, the starting time of activity j in the predictive schedule. This approach is commonly referred to as *railway scheduling*.

Like the basic serial SGS, the *robust serial* SGS considers activities in the order dictated by a priority list, but instead of starting these activities as early as possible, they will be scheduled at their feasible positions that are the closest possible to their planned starting times in the baseline schedule. The deviation $\varepsilon = |s_j^t - s_j^0|$ will thus be minimized for each activity j. Contrary to railway scheduling, this approach allows an activity j to be scheduled earlier than s_j^0. Scheduling activity j at $s_j^0 - \varepsilon$ will even be given priority to scheduling j at $s_j^0 + \varepsilon$ if a tie needs to be broken between both possibilities with equal ε. The non-retroactivity constraint that will be discussed in Section 7.3.1 dictates that at decision point t, an eligible activity should not be scheduled earlier than t. Whenever $\varepsilon > s_j^0 - t$, the robust serial SGS acts like the serial SGS and searches for the earliest resource and precedence feasible starting time for activity j.

7.3 Properties of robust SGSs

In this section we analyze the basic properties of the priority rule-based schedule generation schemes introduced in the previous section. We investigate whether they satisfy the so-called non-anticipativity and non-retroactivity constraints, whether they generate non-delay or active schedules and whether they may suffer from the well-known Graham anomalies (Graham 1966).

7.3.1 Non-anticipativity constraint and non-retroactivity constraint

The *non-anticipativity constraint* (Fernandez & Armacost 1995) specifies that reactive scheduling decisions can only be made on the basis of the observed information from the past and the a priori statistical knowledge of the future. This means that when we have to decide to schedule a set of eligible activities, we do not know how long they will actually take.

As information becomes gradually available during project execution, a scheduling decision made at time t may no longer be the best decision at time $t' > t$. For such cases, a constraint that is complementary to the non-anticipativity constraint will be introduced, i.e. the *non-retroactivity constraint*. This specifies that a reactive procedure may not overrule previous scheduling decisions by scheduling activities in the past. Violations of this constraint may result in infeasible realized schedules.

Assume that we have a project consisting of three activities and a single renewable resource type with constant availability $a = 4$. The activities have expected durations $E(\mathbf{d}_1) = 2$, $E(\mathbf{d}_2) = 2$, and $E(\mathbf{d}_3) = 3$ and resource requirements $r_1 = 2$, $r_2 = 3$, and $r_3 = 2$. Suppose that at time 0 we decided to schedule them by applying a deterministic serial SGS to priority list $\lambda = (1, 2, 3)$ as shown in Figure 7.2. When at time 2, new information becomes available that reveals that activity 1 will take one extra time unit to complete, the schedule of Figure 7.3 would minimize the makespan, but violates the non-retroactivity constraint because activity 3 cannot be scheduled in the past.

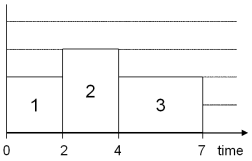


Figure 7.2: Predictive schedule S^0

Clearly, all SGSs described in the previous section respect both the non-anticipativity and the non-retroactivity constraint. Remark that the side constraint added in the stochastic serial SGS to ensure that $s_i \leq s_j$

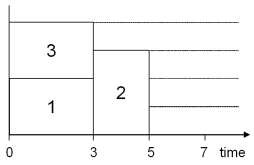


Figure 7.3: Impossible schedule at time 2

for any activity i that precedes activity j in the priority list, also preserves non-retroactivity, but is too strong.

7.3.2 Non-delay and active schedules

A feasible schedule is called *active* if for all activities no local or global left shift can be performed. A feasible schedule is called a *non-delay schedule* if for all activities no local or global left shift can be performed even if activities can be preempted at integer time points. A local left shift of an activity j in a schedule is a left shift that can be obtained by successive one period left shifts of the activity, i.e. all intermediate schedules in which the starting time of activity j is successively decreased by one time unit have to be feasible. A left shift which is not a local left shift is called a global left shift (Brucker & Knust 2006).

Kolisch (1996b) has shown that for the RCPSP with constant resource capacities and regular measures of performance, any schedule generated by the parallel SGS belongs to the set of non-delay schedules and this set possibly does not contain any optimal solution. On the other hand, any schedule generated by the serial SGS belongs to the larger class of active schedules and this class always includes at least one optimal solution for regular performance measures.

The maximum stability objective function (see Eq. 2.3) used in this thesis, however, is a non-regular performance measure. In our problem setting, an optimal solution minimizes the expected difference between the predictive schedule and the realized schedule ($\sum_j w_j E|\mathbf{s}_j^T - s_j^0|$). There is no guarantee for non-regular performance measures that there always exists an optimal non-delay schedule or an optimal active schedule.

Both the robust parallel and robust serial SGS do not necessarily gen-

erate non-delay or active schedules. As with their non-robust variants, the robust serial SGS generates a larger class of schedules than the robust parallel SGS. Still, there does not always exist a priority list λ that would result in the ex-post optimal schedule S^* (see Section 2.1.2.5) even if a static scheduling problem with full information about the realized activity duration is solved. For the robust parallel SGS, an activity will never be started before its baseline starting time, while this might be the case for the ex-post optimal schedule. For the robust serial SGS, an activity is scheduled as close as possible to its baseline starting time. An activity is only scheduled before or after its baseline starting time if scheduling it at its baseline starting time is infeasible.

In short, both robust reactive scheduling schemes try to aggressively repair the predictive schedule and do not intentionally provide any safety cushion against future disruptions. Protecting a schedule against future disruptions is left as the sole responsibility of the proactive scheduling routine.

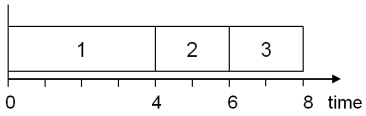


Figure 7.4: Predictive schedule S^0

Given their scheduling characteristics, the robust parallel and robust serial SGS do not necessarily generate optimal reactive schedules. Consider a small project consisting of three activities that can only be scheduled in series as illustrated in the predictive schedule of Figure 7.4. If activity 1 finishes early at time 2, both the robust serial and robust parallel SGS would decide to project activities 2 and 3 at their baseline starting times. Assume that $P(\mathbf{d}_2 = 1) = 0.5$ and $P(\mathbf{d}_2 = 3) = 0.5$. This means that there is a 50% probability that starting activity 2 at its predicted start time $s_2^2 = 4$ induces a one period delay in the start of activity 3 with a cost equal to w_3. The total expected cost would then be $0.5(0) + 0.5(w_3)$. However, if we decide to start activity 2 at time 3, one period earlier than planned, there would have been a 100% probability that we induce a cost w_2, while activity 3 could start as originally planned. If $w_3 > 2 \times w_2$, such a proactive strategy would

result in a lower expected stability cost than the cost obtained by either the robust serial or the robust parallel SGS.

7.3.3 Graham anomalies

Given the type of activity duration uncertainty dealt with in this dissertation, the parallel and serial SGS may suffer from the so-called Graham anomalies (Graham 1966) that were identified for parallel machine scheduling problems under the minimum makespan objective. For example, Figures 7.2 and 7.3 illustrate the anomaly that an activity duration extension may result in a shorter makespan if the serial SGS is used in the deterministic RCPSP setting with priority list $\lambda = (1, 2, 3)$. A one-period duration extension of activity 1 leads to a two-period reduction in makespan.

In this thesis, we are concerned about anomalies that may be induced by activity duration reductions or extensions under the schedule stability objective. Stated otherwise, we are interested to know whether it is possible that an activity duration reduction deteriorates stability or that an activity duration extension improves stability. In order to answer both questions, two cases must be distinguished: (a) the activity disruption is a stand-alone disruption, and (b) other activities are also disrupted (disruption scenario).

7.3.3.1 Stand-alone activity disruption

As long as there is no activity disruption, the projected schedule S^t is, by definition, identical to the predictive schedule S^0 with a stability cost of $\Delta(S^0, S^0) = 0$. Because the stability cost is always non-negative, no activity duration extension can ever decrease the stability cost, independently of the SGS used. This limits our search for stability anomalies to an investigation of the impact of activity duration reductions on stability.

If we apply non-robust SGSs, almost any duration reduction may generate the stability anomaly because a cost is also incurred if an activity starts earlier than its predictive starting time and the successors of the disrupted activity may start earlier.

A robust parallel SGS applies railway scheduling. As a result, a stand-alone activity duration reduction does not affect the time that an activity becomes eligible which equals its starting time in the predictive schedule.

Because the predictive schedule was both precedence and resource feasible, each activity will be scheduled at its baseline starting time, resulting in a schedule with zero stability cost.

The robust serial SGS decides to plan each activity i at its feasible starting time s_i^t with minimal deviation $|s_i^t - s_i^0|$ from its baseline starting time s_i^0. Because a stand-alone duration reduction never prevents an activity i to start at s_i^0, $|s_i^t - s_i^0| = 0$ for each activity and the stability cost will thus again remain zero.

Clearly, in the single activity disruption case, both the robust serial and the robust parallel SGS do not suffer from stability anomalies induced by changes in activity duration.

7.3.3.2 Disruption scenario

It can readily be shown that an activity duration extension (reduction) can improve (deteriorate) stability if other disruptions already occurred.

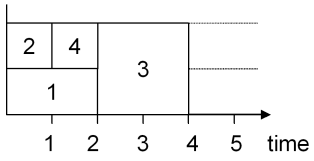


Figure 7.5: Predictive schedule S^0

Let us illustrate this anomaly on a 4-activity project with a single renewable resource type with availability $a = 2$ and resource requirements $r_1 = 1$, $r_2 = 1$, $r_3 = 2$, $r_4 = 1$. Assume that there are no precedence constraints and that the expected activity durations are $E(\mathbf{d}_1) = 2$, $E(\mathbf{d}_2) = 1$, $E(\mathbf{d}_3) = 2$, and $E(\mathbf{d}_4) = 1$. Figure 7.5 shows a baseline schedule with minimum makespan.

Assume that the reactive scheduling schemes use the priority list $\lambda = (1, 2, 3, 4)$. If at time 1, activity 2 has not yet finished, the robust SGSs would both result in the projected schedule S^1 of Figure 7.6, in which only activity 4 does not start at its baseline starting time. The stability cost of this schedule is thus $3 \times w_4$. An extension of activity 1, would result in the projected schedule S^2 of Figure 7.7 at time 2 with a stability cost of $w_3 + w_4$.

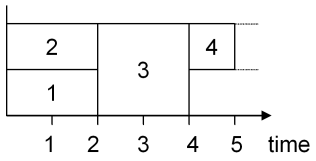


Figure 7.6: Projected schedule S^1 at time 1

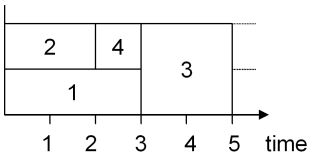


Figure 7.7: Projected schedule S^2 at time 2

It is not difficult to find values for the weights such that this cost will be smaller than the cost of S^1 so that the second activity duration extension would improve stability.

7.3.4 Dispatching

The parallel and robust parallel SGS operate as a *dispatching* or *on-line scheduling* rule, making scheduling decisions dynamically over time. At each decision point, a dispatching rule decides which activities to start without having to decide when the not yet started activities will be projected.

The serial SGS is not commonly used in stochastic scheduling because it does not behave as a dispatching rule. It requires at each decision point t the calculation of a *complete* projected schedule S^t, including best guesses for projected starting times of all the activities that have not yet been started. Full rescheduling at each decision point obviously entails increased computational time. The same reasoning holds for the robust serial SGS. The stochastic serial SGS is based on the serial SGS, but has the advantage that it can be used as a dispatching rule because of its extra constraint that $s_i \leq s_j$ for any activity i that precedes activity j in the priority list.

For a solution robust reactive procedure that tries to minimize the deviation between the projected schedule S^t constructed at current time t and the predictive schedule S^0, having knowledge of the complete projected

schedule is essential. Later in this chapter, the parallel SGSs will be extended so that they allow for a complete projected schedule to be generated at each decision point.

7.4 Reactive scheduling procedures

In this section, several possible reactive procedures are defined. In order to illustrate these reactive procedures, we start from the quality robust schedule of Figure 2.2 (p. 10) with a project due date δ_9 equal to 20 and we explain how each procedure would react when the disruptions presented in Table 7.1 would occur during project execution. Activity weights w_i are also included in this table. For illustrative purposes, these weights differ from the previously introduced weights in Table 2.1.

In accordance with the definitions of Chapter 2, d_i denotes the expected activity duration and $\mathbf{d}_i$ the realized activity duration. Because of the importance of resource constraints in the reactive phase, we will use the resource profile representation of the project throughout this chapter. Figure 7.8 replicates the resource profile for the minimum duration schedule of Figure 2.2 that was previously shown in Figure 2.5.

Table 7.1: Activity duration disruptions for example network

Act i	d_i	$\mathbf{d}_i$	w_i	$successors_i$
Act 0	0	0	0	1, 2, 3
Act 1	4	7	2	4, 7
Act 2	5	5	1	4
Act 3	2	2	3	5
Act 4	4	5	1	6
Act 5	5	3	8	8
Act 6	4	3	1	9
Act 7	2	4	3	9
Act 8	2	3	6	9
Act 9	0	0	38	/

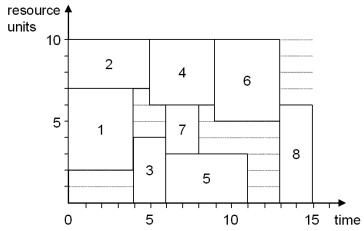


Figure 7.8: Resource profile for the schedule of the example project

7.4.1 Complete rescheduling by resolving the RCPSP

The first reactive procedure studied is to completely regenerate a new up-to-date schedule when schedule breakage occurs. Rescheduling is done by applying the scheduling algorithm that was used to generate the baseline schedule, but now to a modified project network. Activities that are already finished at the schedule breakage point are omitted from the project network. Activities that have already started but did not yet finish by the schedule breakage point are kept in the network with the projected remainder of the activity duration as their planned duration.

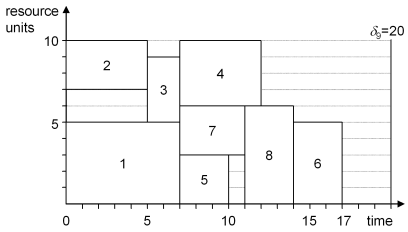


Figure 7.9: Realized schedule obtained by the RCPSP procedure

Figure 7.9 shows the realized schedule that was obtained by applying the branch-and-bound procedure originally used to derive the baseline schedule. It can be observed that activity 8 is scheduled in front of activity 6 in order to finish the project as soon as possible. Activity 6 cannot start at time instant 11 due to the one-period disruption in activity 4, its predecessor in the network. Activity 5 finishes earlier than originally planned, creating the possibility for activity 8 to start at time instant 11. The project finishes

at time 17 with a stability cost of 33.

7.4.2 Fixed resource allocation

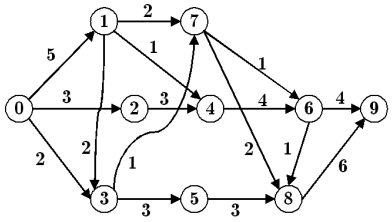


Figure 7.10: Resource flow network for example project

The next reactive scheduling procedure studied applies an early start policy at the schedule breakage point, while maintaining the resource allocation decisions made in the baseline schedule, as reflected in its resource flow network of Figure 7.10 (see also Figure 2.4). As activity duration disruptions do not change the resource requirements of any activity, maintaining the resource flow network that was constructed for the baseline schedule and applying an earliest start policy (Radermacher 1985) upon schedule disruption will yield a precedence and resource feasible projected schedule (see Leus & Herroelen (2004)). This schedule allows every activity to start as early as possible as soon as all its precedence and resource-based predecessors in the resource flow network have finished. Applying an earliest start policy after fixing the resource flows can thus be seen as simple *right shifting* the remainder of the baseline schedule.

For the disruption scenario of Table 7.1 for our problem example, the reactive policy must be applied for a first time at time instant 4 when activity 1 is planned to finish. Because activity 1 suffers from a duration increase of three time periods, activity 2 is the first activity to finish at time 5. The actual activity duration of activity 1 is unknown at time 4, so that activity 1 is scheduled with a remaining duration of one time unit. Maintaining the resource flow of two resource units from activity 1 to 3 (see Figure 7.10) results in the fact that the start time of activity 3 is delayed up to time instant 5. Planning the remaining activities as early as possible for the

given resource flows yields the *projected schedule* of Figure 7.11.

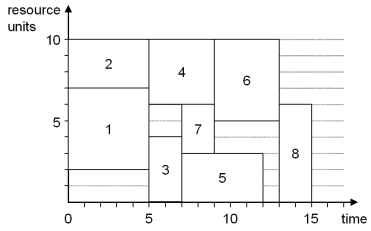


Figure 7.11: Projected schedule generated by the fix flow reactive procedure at time instant 4

Taking into account all the disruptions of the disruption scenario shown in Table 7.1, the final 19-period realized schedule generated by the fix flow reactive procedure is shown in Figure 7.12. Given the activity weights w_i of Table 7.1 the stability cost can be computed as $\sum w_j|\mathbf{s}_j - s_j| = 66$.

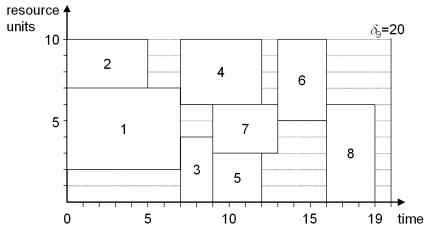


Figure 7.12: Realized schedule obtained by fixing the resource flows

This reactive procedure can be made solution robust by adding railway scheduling. Activities will only become eligible for scheduling at their baseline starting time. For our example scenario of Table 7.1, this solution robust fixed resource allocation would again result in the realized schedule of Figure 7.12 because $\forall i \in N : s_i^t \geq s_i^0$.

7.4.3 Priority lists scheduling

The schedule generation schemes described in Section 7.2 were based on a precedence feasible[2] priority list to decide which activities to schedule at

[2]A priority list λ is precedence feasible if $\forall(i,j) \in A$: i precedes j in λ.

each decision time. Literature on priority lists for the static RCPSP is very extensive. We refer to Kolisch (1996a) and Kolisch & Hartmann (2006) for excellent overviews. We will concentrate on priority lists that are generated in hope of good performance on solution robustness measures. The following *static priority rules* for composing the priority list are proposed:

- EBST = earliest baseline activity starting time
- LST = latest starting time
- LW = largest activity weight
- LAN = lowest activity number
- RND = random

The EBST rule orders the activities in non-decreasing order of their starting times in the predictive schedule S^0. The LST priority rule (Alvarez-Valdes & Tamarit 1989), that orders the activities in non-decreasing order of their latest starting time, is included because it ranks among the best priority rules for the deterministic RCPSP. Ties can be broken by ordering the activities in decreasing order of their weights w_j (EBST1 and LST) or increasing order of their activity numbers (EBST2). The LW rule gives priority to activities with a large disruption cost w_j (smallest activity number as tie-breaker). The LAN rule orders the activities in increasing order of their activity number. The RND rule generates the priority list fully randomly.

Dynamic priority lists depend on the current projected schedule and should thus be updated at each decision time t. The information about the past is known and might influence decisions about the future. We consider two dynamic priority lists:

- EPST = earliest projected starting times
- MC = minimal cost

The EPST rule orders activities at time t by increasing starting times s_i^{t-1} of the projected schedule S^{t-1} generated at the previous decision time.

The MC rule orders the activities by increasing $w_i(s_i^0 - t)$. This value will be negative when $s_i^0 < t$ and positive when $s_i^0 \geq t$. For activities with $s_i^0 < t$, priority is given to activities that induce a high stability cost $w_i|s_i^0 - t|$ if started at time t, because delaying their start to a later starting time would even be worse. On the other hand, among the activities with $s_i^0 \geq t$, we prefer to schedule activities with low stability cost first.

We will illustrate the working principles of the serial SGS on the EBST1 priority list for the disruption scenario of Table 7.1. $\lambda = (0, 1, 2, 3, 4, 5, 7, 6, 8, 9)$ The obtained realized schedule is shown in Figure 7.13. It has a makespan

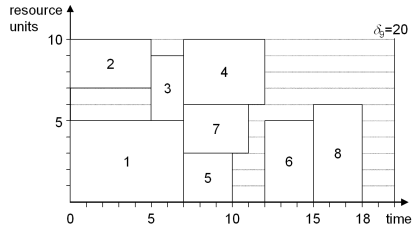


Figure 7.13: Realized schedule obtained by applying the serial robust SGS to λ_{EBST}

of 18 time periods and a stability cost of 31. It differs from the schedule generated by the RCPSP procedure in that activity 6 which precedes activity 8 in the activity list, is scheduled as soon as possible at time instant 12 upon completion of activity 4. This prohibits activity 8 from starting before activity 6, although starting activity 8 before activity 6 results in a better makespan (see Figure 7.9) and lower stability cost (see Figure 7.14). The robust serial SGS would result in the same realized schedule.

7.4.4 Sampling approach

It will typically be observed that no single priority list can be advised for all decision times. A good priority rule for an activity duration decrease will probably differ from a good rule for a duration increase, etc. However, mostly when a priority rule results in a bad decision, the projected schedule constructed at the decision time itself will already indicate the low quality of the decision taken by a high value for $\Delta(S^t, S^0)$.

This section describes two reactive scheduling sampling schemes that

rely on several priority lists in combination with several schedule generation schemes: *basic sampling* and *time-window sampling*. Sampling in this context means that at any decision time several feasible solutions are generated and evaluated and that the best candidate solution is selected.

7.4.4.1 Basic sampling

The basic sampling approach shown in Algorithm 7.1 tries to make a suitable scheduling decision at any decision time t as follows:

Algorithm 7.1 Basic sampling

for $t = 0, \ldots, T$ **do**
Step 1: Check for new scheduling information.
Step 2: **If** no new information **then** $S^t = S^{t-1}$ and **goto** period $t+1$
else goto step 3
Step 3: **For** list $l = 1, \ldots, L$ **do**
Construct $S^t_{\lambda_l,RP}$ and calculate $\Delta(S^0, S^t_{\lambda_l,RP})$
Construct $S^t_{\lambda_l,RS}$ and calculate $\Delta(S^0, S^t_{\lambda_l,RS})$
Construct $S^t_{\lambda_l,P}$ and calculate $\Delta(S^0, S^t_{\lambda_l,P})$
Construct $S^t_{\lambda_l,S}$ and calculate $\Delta(S^0, S^t_{\lambda_l,S})$
Store the projected schedule S^t that minimizes $\Delta(S^0, S^t)$
Step 4: Start all activities i with $s^t_i = t$.

Step 1 checks for new information becoming available at time t. If at time t, no activity finishes and no activity was projected to finish, then no new information is available compared to the previous decision point $t-1$. The previous projected schedule S^{t-1} remains valid (Step 2).

Instead of using one priority list in combination with one SGS, Step 3 uses multiple lists $\lambda_l \in \{\lambda_1, ..., \lambda_L\}$ at time t in combination with several SGSs. For each of these lists λ_l, a complete projected schedule is constructed using the robust parallel SGS ($S^t_{\lambda_l,RP}$), the robust serial SGS ($S^t_{\lambda_l,RS}$), the parallel SGS ($S^t_{\lambda_l,P}$) and the serial SGS ($S^t_{\lambda_l,S}$). Doing so, a total of $4 \times L$ candidate projected schedules are generated, with L identifying the number of lists. The projected schedule S^t that accounts for the smallest deviation $\Delta(S^0, S^t)$ among them is stored. The procedure continues in Step 4 by

starting the activities i that have projected starting times $s_i^t = t$ in S^t. Remark that in order to compare the stability costs of all candidate solutions, a complete projected schedule must always be made at any decision time by using the statistical knowledge of future activities. This means that the dispatching advantage of the parallel SGSs cannot be exploited.

7.4.4.2 Time-window sampling

The main problem that may occur in the standard sampling approach is that the decision whether a candidate projected schedule is selected or not may depend on activities that are projected much later in the project at time t'. However, these projected activities are still subject to major uncertainties (*non-anticipativity constraint*) and should not predominate the current decision process. There is no reason to assume that the current reactive policy will also be applied at time t'.

Time-window sampling (TW sampling) tries to cope with this problem by making use of a time window (TW). The difference with the basic sampling approach lies in the generation of the candidate projected schedules $S^t_{\lambda,\cdot}$ at time t (Step 3). Instead of generating a complete projected schedule by applying the current SGS to the current list λ, TW sampling only applies this policy to decide which activities to project within a certain time window $[t, t+\Theta]$. Activities that are not planned to start within this time window, are projected by following the priority list $\lambda_1 = EBST1$. The generation scheme to transform this priority list into a projected schedule is the robust variant of the SGS applied within the time window. Note that deciding which activities to project within $[t, t+\Theta]$ already requires a complete schedule if we apply a non-dispatching SGS such as the serial SGS. For on-line scheduling procedures, such as the parallel SGS, a complete projected schedule is only required once. The dispatching advantage of the parallel SGSs that was absent in the basic sampling procedure can be exploited in TW sampling.

The results of basic and TW sampling depend on the number of priority lists L used and the actually selected priority lists λ_l. For TW sampling the time window size Θ will be an important parameter. Parameter settings will be discussed in Section 7.5.2.

7.4.5 Solving the weighted earliness-tardiness problem

The reactive scheduling problem at each decision point can be viewed as a *Resource-Constrained Project Scheduling Problem with Weighted Earliness-Tardiness Costs* (problem *m,1|cpm|early/tardy* in the notation of Herroelen et al. (2000)). Due dates are set equal to the activity completion times $s_j^0 + E(\mathbf{d}_j)$ in the predictive schedule. The earliness and tardiness costs will be symmetrical and used as the weights w_j in the stability objective function, except for the earliness cost of the dummy end activity which will be set equal to zero.

Some efficient exact procedures for solving problem *m,1|cpm|early/tardy* have been proposed in the scheduling literature (Schwindt 2000, Vanhoucke et al. 2001, Kéri & Kis 2005). The application of such an exact procedure at any decision time for the example problem and the disruption scenario of Table 7.1 yields the 19-period realized schedule of Figure 7.14 with a stability cost equal to 23. The resource flow at time instant 5 is changed. Since activity 1 suffers from a three-period duration increase, activity 3 will no longer wait for activity 1 to finish, but will instead receive its resources from activity 2. In doing so, activity 3 only starts one time period later than originally planned. Also, the heavily weighted activity 8 ($w_8 = 6$) now jumps in front of activity 6 ($w_6 = 1$), so that activity 8 can start as originally planned. Observe that all resource units are kept idle between time 12 and time 13.

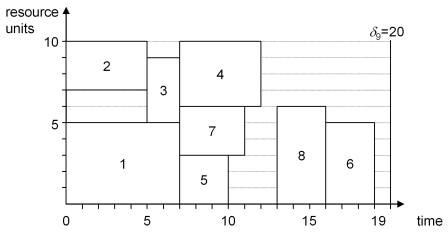


Figure 7.14: Realized schedule generated by the WET procedure

However, Van de Vonder et al. (2005b) showed that calling an exact weighted earliness-tardiness procedure at any schedule breakage point is already computationally infeasible for small network sizes. Recently,

Ballestín & Trautmann (2006) developed a population-based *iterated local search* (ILS) metaheuristic (Lourenço et al. 2002) for the weighted earliness-tardiness resource-constrained project scheduling problem with minimum and maximum time lags (problem *m,1|gpr|early/tardy*. Although faster than exact procedures, this procedure still remains computationally demanding for our problem. Our adapted version of the algorithm described in Ballestín & Trautmann (2006) is customized to the special characteristics of the problem by ommitting some of the original features of the algorithm to reduce the computational requirements . We refer to Ballestín & Trautmann (2006) for a computational evaluation of the iterated local seach algorithm compared to truncated versions of the branch-and-bound codes of Schwindt (2000) and Vanhoucke et al. (2001).

The algorithm runs as follows:

Algorithm 7.2 WET_Procedure

1. Initialize Elite Set.
2. **While** without_imp < max_without_imp **do**
   a. Select a schedule S from Elite Set.
   b. S' = Perturbation2(S).
   c. S" = Local_Searches(S').
   d. **If** $\Delta(S'', S^0) < \Delta(S^+, S^0)$ **then** {without_imp = 0, $S^+ = S''$}.
   e. **Else** without_imp = without_imp+1.
   f. **If** $\Delta(S'', S^0) < \Delta(S^-, S^0)$ **then** S" replaces S^- in Elite Set.
3. Return the best solution obtained: $S^t = S^+$.

S^+ and S^- are the best and worst solution of Elite Set respectively.

Step 2 is repeated iteratively until the number of iterations without improvement (without_imp) equals a predefined number (max_without_imp). Define *card* as the cardinality of the Elite Set. At any time, the *card* best solutions are stored. We initialize the Elite Set in Step 1 as described in Algorithm 7.3.

In Perturbation1, *nunsche*/10 activities are subsequently chosen and reintroduced in the activity list elsewhere, where *nunsche* is the number of unscheduled activities. The swaps are made so that the outcome is a feasible priority list. Perturbation2 delays the position of some advanced activities

Algorithm 7.3 Initialize Elite Set(*card, nitial, nitial2*)

1. $S = S^t_{\lambda_{EBST1},RS}$. $S2 = S^t_{\lambda_{EPST},RS}$.
2. S' = Local_Searches(S). S2' = Local_Searches($S2$).
3. Temporary Set = {S',$S2$'}. Elite Set ={ }.
4. **For** $i = 0, i <$ *nitial2* $- 2$ **do**
   a. **If** i is even **then** λ = Perturbation1(λ_{EBST1}).
   b. **Else** λ = Perturbation1(λ_{EPST}).
   c. Construct $S_i = S^t_{\lambda,RS}$ and calculate $\Delta(S^0, S_i)$.
5. Temporary Set = Best *nitial* solutions from the *nitial2* initial solutions.

For $i = 0, i <$ *nitial* **do**
   a. Restore S_i from Temporary Set
   b. S' = Local_Searches(S_i).
   c. Elite Set = Elite Set $\cup$ {S'}.

6. Elite Set = Best *card* solutions from the *nitial* initially generated solutions.

and advances the position of some delayed activities in the activity list. It is described in more detail under the name of Perturbation4 in Ballestín & Trautmann (2006). Local_Searches includes three of the Local Searches used in that paper. The second one sorts the early activities in decreasing order of their finish times in the solution S and schedules each activity as closely as possible to its due date. Then it proceeds analogously with the delayed activities, sorting them in increasing order of their starting times in S. Before applying this local search, we employ LocalSearch 1, which allows more freedom in the movement of activities. Each early (delayed) activity is scheduled as late (early) as possible, but the movement is stopped right before the objective function deteriorates. For each delayed activity that can be left shifted due to precedence relationships, the last local search calculates the set of activities B that restrain this movement and that can be moved. The function considers each activity $j \in B$ and unschedules it. Then it calculates whether it is better to schedule first i and then j, each of them as close as possible to its real due date. After working with all the activities of B, the method performs the best of these movements if it produces an improvement in the objective function.

7.5 Experimental results

We refer to the paper by Van de Vonder et al. (2006a) for an extensive discussion of computational results. All solution robust reactive procedures have been coded in Microsoft Visual C++ 6.0 and have been tested on the 600 120-activity instances of the well-known PSPLIB 120 data set (Kolisch & Sprecher 1997). Computational results are highly dependent on parameter settings. Rather than repeating them, we will dwell on the major conclusions drawn in that paper. Some of the best performing procedures will be included in the experimental design of Chapter 8. The fixed resource allocation procedure of Section 7.4.2 will then serve as benchmark procedure.

7.5.1 Results obtained by the priority policies

The performance of applying the robust SGSs of Section 7.2 to a priority list λ depends on the used priority list. We remark that the priority list should be ordered by increasing starting times (EBST1, EBST2 or EPST) to obtain good results. Despite the extra computational time spent by dynamic lists to constantly update the priority list based on the new information acquired in the current projected schedule, they do not obtain better results. After all, we try to minimize the expected deviation between $\mathbf{S}^T$ and the predictive schedule S^0, not the deviation between $\mathbf{S}^T$ and S^t. Applying the robust parallel SGS on the list λ_{EBST1} obtains the best overall results. The robust serial SGS obtains very similar results on λ_{EBST1}, but the results of this SGS are more sensitive to the priority list that is applied. Static priority lists combined with a (robust) parallel SGS are also computationally the most efficient procedures among the priority policies and will thus be included in the computational experiment of Chapter 8. The conclusions of this section hold when both a quality robust or a solution robust schedule (STC schedule of Section 5.1.3) are used.

7.5.2 Results of sampling and ILS

Next we will investigate the results obtained by the more advanced heuristics described in this chapter, being basic sampling, TW sampling and ILS.

The performance of the sampling procedure highly depends on the lists λ_l that we include. We experimentally decided to include the six static priority lists of Section 7.4.3 and their precedence feasible backward lists (BW). A backward list is a precedence feasible priority list with reverse priorities for the activities. The backward list of the EBST rule is thus the Latest Baseline Starting Time rule, etc. These backward lists might seem illogical as robust reactive procedures but will prove their use in the sampling procedure. The logic behind their inclusion is that when a priority list does not result in a schedule with low deviation from S^0, its inverse list might do so. The backward list of λ_{EBST2} has been removed from consideration because it hardly improved any results due to its resemblance with *EBST1 BW*. The number of included lists L thus equals 11, resulting in $4 \times L = 44$ candidate projected schedules at any decision time. When the baseline schedule is a solution robust buffered schedule, the candidate schedules obtained by applying non-robust SGSs are excluded from consideration.

Table 7.2: Selection frequency of all policies by *sampling*

λ	RP	RS	P	S	Total
1. EBST1	37.05%	23.16%	1.11%	2.72%	64.03%
2. EBST2	4.94%	5.81%	0.40%	1.33%	12.48%
3. LAN	4.99%	1.02%	0.21%	0.06%	6.28%
4. LST	2.06%	0.45%	0.09%	0.01%	2.61%
5. LW	2.79%	0.54%	0.13%	0.03%	3.49%
6. RND	1.99%	0.32%	0.09%	0.01%	2.42%
7. EBST1 BW	1.56%	0.33%	0.09%	0.02%	1.99%
8. LAN BW	1.35%	0.23%	0.08%	0.02%	1.68%
9. LW BW	1.35%	0.25%	0.07%	0.01%	1.67%
10. RND BW	2.05%	0.37%	0.09%	0.02%	2.52%
11. LST BW	0.65%	0.12%	0.04%	0.01%	0.82%
Total	60.78%	32.61%	2.39%	4.23%	

Basic sampling obtains a substantial stability improvement compared to the best single priority list policies (see also Chapter 8). In most cases

several policies will result in projected schedules that are either identical or have equal stability cost. The ordering of the policies is important because a policy will only be selected if its projected schedule has a lower stability cost than all previously considered schedules. We start by considering all projected schedules generated by the robust parallel SGS, ordered by increasing numbers as indicated in Table 7.2. Afterwards the robust serial, parallel and serial SGS will be considered respectively. Table 7.2 shows the frequencies that each policy is actually selected as best by the sampling procedure for repairing the schedules obtained by the procedure of Debels & Vanhoucke (2006) on the PSPLIB J120 data set. At most decision points (64%), any of the schedules that uses EBST1 as priority list is selected. More importantly, however, is that also the overall less performing policies, such as the non-robust ones (aggregated selection frequency of almost 7%) and the ones that follow backward priority lists (aggregated selection frequency of almost 9%) were selected quite often. Excluding them would deteriorate results substantially.

TW sampling (Section 7.4.4.2) yields better results than basic sampling, but requires an extra parameter setting. There is no single time window Θ size that performs well for all project instances. Taking Θ too small would imply too few differences between the candidate projected schedules and taking Θ too large would result in the disadvantages of basic sampling. Because the serial SGSs require the construction of two complete schedules at any decision time, i.e. one to decide which activities to start in $[t, t + \Theta]$ and one to complete the projected schedule, TW sampling requires more computational time than basic sampling.

The trade-off between performance and computational requirement of the Iterated Local Search (ILS) procedure also depends highly on the parameter settings. In Van de Vonder et al. (2006a), we experimentally set $card = 3$, $nitial = 10$ and $nitial2 = 50$. For these parameters, ILS obtained slightly better results than sampling, but required more than three times the computational time. For project managers that rely on simulation methods to evaluate the predictive schedule (see Chapter 9) of their project, more computationally demanding metaheuristics or exact WET procedures are obviously not suitable as reactive scheduling policies.

ILS generally needs some time to substantially improve results compared to the priority list policies. At first, all priority lists in the Elite Set will be situated in the neighborhoods of EBST1 and EPST1. Sampling directly compares totally different solutions and will improve stability even if the number of lists L is very small. On the other hand, the performance of local search algorithms improves when more time is allocated, while improving the sampling results by adding more lists to the current set of L lists is challenging.

7.6 Conclusions of this chapter

In this chapter we examined several reactive procedures to repair a project schedule whenever activities are disrupted during execution. We observe that after initial schedule selection (Chapter 6), resource allocation (Chapter 4) and buffer insertion (Chapter 5), clever reactive scheduling is a fourth technique to effectively increase the stability of a project. All major parts of Figure 2.6 have now be discussed. The findings of this and previous chapters will be examined in the large computational experiment of the next chapter. Afterwards, a last chapter will translate this work into a real-life environment.

We assumed that protecting a schedule against future disruptions is the sole responsibility of the proactive scheduling routine. In the planning phase, we construct a proactive predictive schedule that will anticipate most future disruptions. When this plan becomes infeasible during project execution, we try to repair this predictive schedule in the best possible way. None of these procedures has a proactive nature. Opportunities to include extra safety are not exploited. Developing reactive procedures that anticipate future disruptions is a promising future research direction.

A second future research topic might be to apply a sampling approach to stochastic scheduling, i.e. to minimize the expected makespan. This would require to select at every decision time the candidate projected schedule with minimum makespan, rather than the one with minimum deviation from the predictive schedule. Different priority lists should be incorporated to do so.

Chapter 8

An experimental analysis of proactive-reactive project scheduling

As we stated in Chapter 1, mainly two approaches exist to induce robustness into a project, i.e. *proactive* and *reactive scheduling*. Chapters 4, 5 and 6 introduced several proactive procedures to create robust baseline schedules. Afterwards, Chapter 7 was dedicated to the investigation of several reactive procedures that dictated how to revise or repair a schedule when an unexpected event occurs.

The current chapter combines both approaches in a large-scale experimental analysis. We tackle the problem in a roundabout way. Solutions to the problem will be obtained through the application of different predictive-reactive scheduling procedures from the previous chapters. Solutions will be evaluated by their value on the objective function of Eq. 2.3, which minimizes the expected deviation between the activity starting times in the baseline schedule and the realized schedule. The analytic evaluation of this solution robustness objective function is very cumbersome (the PERT problem is $\sharp P$ complete (Hagstrom 1988)). Therefore the objective function values will be evaluated through simulation.

The simulation runs will allow us to identify under which conditions proactive scheduling pays off. They will also enable an analysis of the impact

of (a) the level of variability in the activity durations, (b) the relative weights of the project activities and (c) the tightness of the project due date.

The remainder of this chapter is organized as follows. The next section represents an overview of the proactive and reactive scheduling procedures included in the experiment. Section 8.2 is devoted to a description of the experimental set-up used in the computational experiment. The computational results are described in Section 8.3. A last section provides some overall conclusions.

8.1 Proactive-reactive scheduling procedures

Initial schedule selection (Chapter 6), solution robust resource allocation (Chapter 4), buffer insertion (Chapter 5) and reactive scheduling (Chapter 7) have all been shown to be able to improve the stability of a project. Rather than giving a complete recapitulation of all procedures introduced in previous chapters, we aim to compare and combine some of the most promising among them. The magnitude of the improvements obtained by the approaches of the different chapters will be compared. We introduce a classification scheme to describe the different proactive-reactive scheduling combinations by using four fields, i.e. τ, υ, ϕ and χ such that any predictive-reactive scheduling strategy x can be classified as $|\tau_x|\upsilon_x|\phi_x|\chi_x|$ where τ_x, υ_x, ϕ_x and χ_x are values for the corresponding classification fields. Sections 8.1.1, 8.1.2, 8.1.3 and 8.1.4 are assigned to descriptions of these four parameter fields and reveal which procedures will be used in the experiments of this chapter.

8.1.1 Initial schedule selection (τ)

Buffer insertion techniques add safety to an initial unbuffered schedule S^U. The importance of the initial schedule choice has been demonstrated in Chapter 6. The first field τ of our four-field classification scheme indicates the initial schedule procedure. $\tau = DV$ denotes that we generate a single initial schedule by applying the combined crossover algorithm deve-

loped by Debels & Vanhoucke (2006)[1] to solve the deterministic RCPSP with $d_j = E(\mathbf{d}_j)$. A total of 50,000 schedule generations is applied as stop condition. This algorithm has been shown to be among the best performing metaheuristic RCPSP procedures for both small and larger networks.

$\tau = BEST$ signifies that we search for a solution robust initial schedule, rather than a random one. The algorithm applied when $\tau = BEST$ depends on the data set used. For the PSPLIB J30 data set we use the branch-and-bound code of Section 6.1 with $\Delta = 0$ to enumerate all possible minimum duration schedules. We truncate the procedure whenever either 10,000 candidate schedules have been found or whenever a new solution has been found and the procedure runs for over 10 minutes. All candidate schedules are then evaluated by simulation and the best one is selected as S^U. It should already be remarked that such candidate evaluation depends on the buffer insertion method and on the reactive procedure. These interrelations will be dealt with later in this chapter.

For the PSPLIB J120 data set, $\tau = BEST$ denotes that the initial schedule has been generated by applying metaheuristic 2, which was described in Algorithm 6.4. Note that the resulting initial schedule is already buffered in this case.

8.1.2 Resource allocation (υ)

The MABO heuristic ($\upsilon = MABO$) of Section 4.2.2 will be used as a solution robust resource allocation procedure. This decision has been authorized by the conclusions made in Deblaere et al. (2006) and in Chapter 4 of this dissertation. The simple procedure of Artigues et al. (2003) will be used as a benchmark procedure ($\upsilon = ART$).

8.1.3 Buffer insertion (ϕ)

Two buffer insertion methods will be included in the factorial designs later in this chapter and they will be indicated by the classification field ϕ. First, when no buffers ($\phi = NONE$) are added to S^U, a quality robust

[1] An executable program can be downloaded at the following website: www.projectmanagement.ugent.be/downloads/RCPSP

baseline schedule is obtained. Second, the time-consuming improvement heuristic STC D ($\phi = STC\ D$) (see Section 5.1.4) is used to generate a baseline schedule with excellent solution robustness. Although the STC heuristic ($\phi = STC$) of Section 5.1.3 is not included in the factorial designs of this chapter, we will occasionally refer to the results obtained by this heuristic, which has been shown to be the best among the simple heuristics to generate solution robust baseline schedules.

As we already stated in Section 8.1.1, the evaluation of candidate initial schedules relies on the buffer insertion method that has been applied. When the baseline schedule is buffered ($\phi \neq NONE$), we concluded in Section 6.1 that it was highly suggested to evaluate the initial schedules after buffer insertion. However, evaluating the candidate solutions by applying a computationally heavy procedure such as STC D to all of them is too demanding. Hence, when $\phi = STC\ D$ the candidate solutions will be evaluated by simple STC and the improvement phase will only be executed once to transform the selected initial schedule S^U into the baseline schedule S^0.

8.1.4 Reactive procedures (χ)

The last field indicates the reactive procedure applied whenever disruptions occur during (simulated) project execution. Keeping the resource flow allocations decided by one of the algorithms of Section 8.1.2 fixed, will serve as our benchmark reactive procedure ($\chi = FF\ R$). The R denotes that railway scheduling is applied in this approach. Reactive procedures without any form of railway scheduling will not be considered because their principles are contradictory to the safety added in buffered schedules.

The basic sampling approach ($\chi = SAMP$) of Section 7.4.4 is included as a reactive procedure that obtains excellent results on stability. The parameter settings for the sampling approach are as discussed in Section 7.5.2. Applying the robust parallel SGS on the priority list λ_{EBST1} scored best on solution robustness among several priority rule policies in the previous chapter and will thus be used under the classification $\chi = EBST$. However, like $\phi = STC$, EBST is not included in the factorial designs that will appear later in this chapter.

The branch-and-bound procedure for initial schedule selection and the

STC D buffer insertion algorithm both rely on reactive procedures to evaluate several candidate solutions by simulation. It is computationally too demanding to apply the sampling approach of Section 7.4.4 to each candidate solution. The most often selected priority rule policy during sampling, i.e. applying the robust parallel SGS to λ_{EBST1}, will be applied instead.

The MABO procedure also selects the best among multiple alternatives by evaluating them through simulation. However, deciding on the best resource allocation is always done by assuming that these resource allocation decisions would be kept fixed during project execution, whatever the actual reactive procedure (see χ) during project execution is.

8.2 Experimental set-up

The predictive and reactive scheduling procedures were coded in Microsoft Visual C++ and tested on the well-known J30 and J120 PSPLIB data sets (Kolisch & Sprecher 1997). For details on these instances we refer to the parameter settings section of the PSPLIB website[2]. In this computational experiment we will study the impact of three parameters: the level of uncertainty in activity durations, the weighting parameter and the project due date.

In order to investigate the impact of activity duration variability, we assume that an activity duration can have low, medium and high duration variability. *High duration variability* means that the real activity durations are all discretized values drawn from a right-skewed beta-distribution with parameters 2 and 5, that is transformed in such a way that the minimum duration equals $0.25 \times E(\mathbf{d}_j)$, the mean duration equals $E(\mathbf{d}_j)$ and the maximum duration equals $2.875 \times E(\mathbf{d}_j)$. *Low duration variability* means that the realized activity durations are also discretizations of values drawn from a beta-distribution with parameters 2 and 5, but with the mean equal to $E(\mathbf{d}_j)$ and with minimum and maximum values equal to 0.75 times and 1.625 times $E(\mathbf{d}_j)$, respectively. *Medium duration variability* is an intermediate case where the realized activity durations are drawn from a beta-distribution with parameters 2 and 5, but with minimum and maximum values equal to

[2]http://129.187.106.231/psplib/

0.5 times and 2.25 times $E(\mathbf{d}_j)$, respectively. Figure 8.1 shows the distribution functions from which the realized durations are drawn for an activity with an expected 3-period duration.

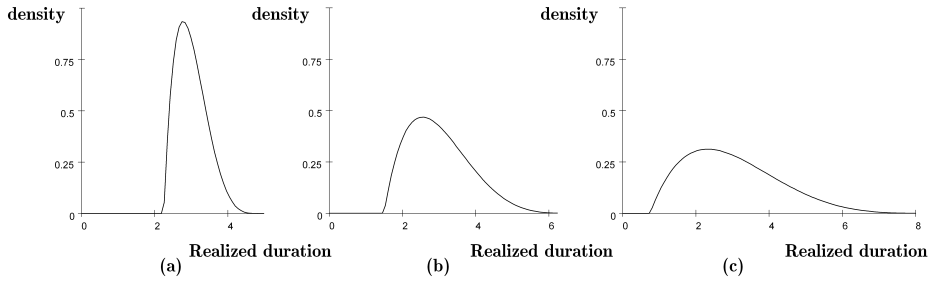


Figure 8.1: Distribution functions for low (a), medium (b) and high (c) duration variability if $E(\mathbf{d}_i) = 3$

We assume that the variabilities are activity dependent and distinguish between two overall uncertainty levels for projects. In a highly uncertain environment, we randomly select for every activity whether the activity has *high*, *low* or *medium duration variability*. If the overall uncertainty level is low, we randomly select for each activity whether it has *low* or *medium* duration variability. Highly uncertain activities do not occur in such environments.

The *weighting parameter* (wp) was defined in Section 2.3 as the ratio between the weight of the dummy end activity and the average of the distribution of the weights of the other activities ($wp = w_n/w_{avg}$). The activity weights w_j of the non-dummy activities $j \in \{1, 2....n-1\}$ are drawn from a discrete triangular distribution with

$$P(w_j = q) = (21 - 2q)\% \qquad \forall q \in \{1, 2....10\} \tag{8.1}$$

This distribution results in a higher probability for low weights and in an average weight $w_{avg} = 3.85$ that is used to calculate $w_n = wp \times w_{avg}$. In this experiment, two values for wp ($wp = 5$ and $wp = 10$) are examined.

We distinguish between two possible settings for the *project due date*: a 10% and a 30% increase above the makespan of the initial schedule that solves the deterministic RCPSP with mean activity durations as described

in Section 8.1.1. Following the experimental output of Chapter 3, these two project buffer sizes can be regarded as rather tight and loose.

8.3 Computational results

All computational results have been obtained on a Pentium IV 2.4 GHz personal computer. For each of the 480 PSPLIB J30 and 600 PSPLIB J120 project network instances two data sets (referred to further on as the *training set* and the *test set*) of both 100 project executions are simulated by drawing beta-distributed activity durations as discussed above. The test set of executions is run to detect *overfitting* as will be explained below. It should be recognized that for smaller networks, more simulation runs should be executed to obtain completely converging results, i.e. Leus (2003) shows that solutions on 30-activity networks only converge after 350 iterations. However, all procedures in the experimental analysis are tested on the same sets of 100 simulated scenarios for each network, such that their comparison on an independent test set does not require individual convergence to allow for the observance of significant differences between the results of several proactive-reactive procedures on the training and the test set.

The rows of Tables 8.1 and 8.2 show the average results obtained by a particular predictive-reactive procedure $|\tau_x|\phi_x|\upsilon_x|\chi_x|$ on PSPLIB J30 and PSPLIB J120 respectively. For the results of Tables 8.1 and 8.2, the due date has been set at a 30 % increase compared to the minimum makespan, we assume a highly variable project environment as described above and $wp = 10$. The impact of these parameter settings will be discussed later.

The column with heading *Training* shows the stability cost $\sum w_j E\left|\mathbf{s}_j - s_j\right|$ averaged over all network executions in the training set. The column with heading *Test* shows the average stability cost for the disruptions of the test set. A last column denotes the average computational time in seconds required per network (for 100 simulated executions).

Row 1 indicates the stability results when no effort is done to obtain good solution robustness. The CPU time is almost completely consumed by solving the RCPSP. The resource allocation procedure $\upsilon = ART$ and the reactive procedure $\chi = FF\ R$ run in virtually no time.

Table 8.1: Performance values on PSPLIB J30

Row Nr.	τ	υ	ϕ	χ	Training	Test	Time
1	DV	ART	NONE	FF R	340.78	341.53	0.39
2	DV	ART	NONE	SAMP	250.76	251.64	2.19
3	DV	ART	STC D	FF R	43.64	47.42	0.61
4	DV	ART	STC D	SAMP	29.90	33.12	3.91
5	DV	MABO	NONE	FF R	299.33	300.69	0.40
6	DV	MABO	NONE	SAMP	250.76	251.64	2.20
7	DV	MABO	STC D	FF R	35.73	39.27	0.62
8	DV	MABO	STC D	SAMP	29.26	32.36	4.11
9	BEST	ART	NONE	FF R	311.10	314.12	58.98
10	BEST	ART	NONE	SAMP	233.51	235.21	68.96
11	BEST	ART	STC D	FF R	37.7	42.26	70.22
12	BEST	ART	STC D	SAMP	27.37	30.88	78.08
13	BEST	MABO	NONE	FF R	273.00	276.43	73.52
14	BEST	MABO	NONE	SAMP	233.66	235.38	83.72
15	BEST	MABO	STC D	FF R	31.58	35.42	86.94
16	BEST	MABO	STC D	SAMP	26.80	30.04	94.95

Row 16 denotes the performance of the procedure that could be regarded as the most solution robust among all examined procedures. On PSPLIB J30, the average stability cost on the test set has been reduced from 341.53 to 30.04 compared to the initial situation of row 1. The difference between those two extreme cases will be called the *stability gap GP*. For PSPLIB 120, the stability gap equals 4901.77. The time required to obtain this stability improvement is 94.95 seconds for the J30 data set and 172.04 seconds for the J120 data set. For J30 most of it (i.e. 91.25 s) is consumed by the branch-and-bound code to obtain several candidate initial schedules. As already mentioned in Chapter 5, the computational time required by the branch-and-bound code for initial schedule selection is highly variable. 253 out of 480 network instances need less than 1 second to enumerate all minimum duration schedules, while for 19 instances the procedure has been truncated after searching for more than 10 minutes. For the larger networks

Table 8.2: Performance values on PSPLIB J120

Row Nr.	τ	υ	ϕ	χ	Training	Test	Time
1	DV	ART	NONE	FF R	4998.83	5012.37	1.98
2	DV	ART	NONE	SAMP	2217.82	2229.99	24.97
3	DV	ART	STC D	FF R	356.10	391.33	12.90
4	DV	ART	STC D	SAMP	101.48	112.89	105.33
5	DV	MABO	NONE	FF R	4185.09	4218.05	2.85
6	DV	MABO	NONE	SAMP	2217.82	2229.99	25.84
7	DV	MABO	STC D	FF R	236.66	266.30	13.53
8	DV	MABO	STC D	SAMP	98.02	109.78	113.12
9	BEST	ART	NONE	FF R	4866.14	4884.45	62.31
10	BEST	ART	NONE	SAMP	2109.85	2122.08	84.68
11	BEST	ART	STC D	FF R	813.74	859.50	72.93
12	BEST	ART	STC D	SAMP	101.16	113.23	153.47
13	BEST	MABO	NONE	FF R	4025.39	4057.09	63.80
14	BEST	MABO	NONE	SAMP	2109.85	2122.08	86.18
15	BEST	MABO	STC D	FF R	227.22	262.83	72.80
16	BEST	MABO	STC D	SAMP	97.31	110.60	172.04

of PSPLIB J120, the improvement heuristic for buffer insertion (97.18 s), the reactive sampling procedure (11.06 s) and the metaheuristic of Algorithm 6.4 (62.30 s), which is used to generate a solution robust initial schedule for the improvement heuristic, all require substantial computational time. Solution robust scheduling is obviously no easy task for large networks.

Figure 8.2 illustrates the trade-off between solution robustness and required CPU time for the J120 data set. The X-axis denotes the percentage of the stability gap that has been achieved by the different methods. This can be calculated for the proactive-reactive procedure of row x as:

$$\frac{Stability_1 - Stability_x}{GP} \tag{8.2}$$

in which $Stability_x$ is the objective function value obtained after simulation on the test set for procedure x. The results of the 16 combinations are shown on the scatter diagram with the row number in Table 8.2 as label. The results of metaheuristics 1 and 2 that were found in Table 6.1 are also included with

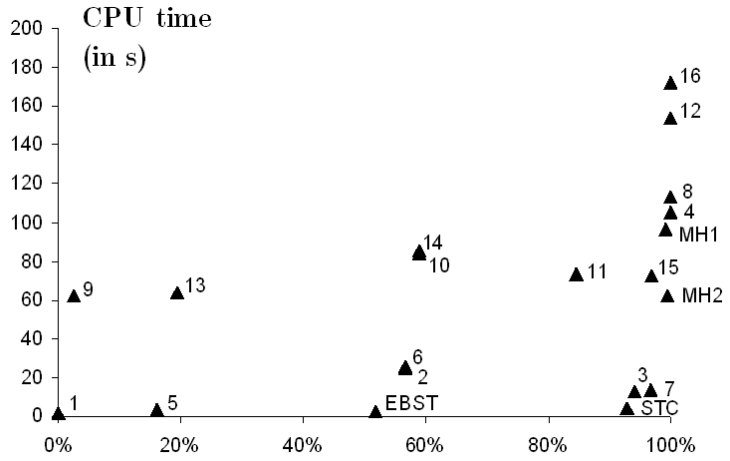


Figure 8.2: Trade-off between solution robustness and required CPU time

labels MH1 and MH2 respectively. Furthermore, the labels STC and EBST refer to the procedures that can be classified under $|DV|ART|STC|FF\ R|$ and $|DV|ART|NONE|EBST|$ respectively.

To effectively analyze the results of Tables 8.1 and 8.2, we perform a full factorial design with four factors that represent the different robustness techniques included in our classification scheme. Each factor has a low and a high level, which correspond to a quality robust and a solution robust procedure respectively. This results in a 2^4 factorial design with 16 predictive-reactive scheduling combinations. The regression model statement of our factorial design looks as follows:

$$\begin{aligned} Y = \;& b_0 \\ + \;& b_\tau X_\tau + b_\upsilon X_\upsilon + b_\phi X_\phi + b_\chi X_\chi \\ + \;& b_{\tau\upsilon} X_\tau X_\upsilon + b_{\tau\phi} X_\tau X_\phi + b_{\tau\chi} X_\tau X_\chi + b_{\upsilon\phi} X_\upsilon X_\phi + b_{\upsilon\chi} X_\upsilon X_\chi + b_{\phi\chi} X_\phi X_\chi \\ + \;& b_{\tau\upsilon\phi} X_\tau X_\upsilon X_\phi + b_{\tau\upsilon\chi} X_\tau X_\upsilon X_\chi + b_{\tau\phi\chi} X_\tau X_\phi X_\chi + b_{\upsilon\phi\chi} X_\upsilon X_\phi X_\chi \\ + \;& b_{\tau\upsilon\phi\chi} X_\tau X_\upsilon X_\phi X_\chi, \end{aligned} \tag{8.3}$$

in which each factor i is represented by a dummy variable X_i, with values -1 and +1 representing the low and high level respectively. The parameters b_i denote the main effects of these dummy variables, while the remaining pa-

Technique	Field	Coefficient	T-statistic	Significant?
Intercept		$b_0 = +156.09$	$t_0 = +185.38$	YES
Initial schedule	τ	$b_\tau = -6.12$	$t_\tau = -7.27$	YES
Resource allocation	υ	$b_\upsilon = -5.93$	$t_\upsilon = -7.04$	YES
Buffer insertion	ϕ	$b_\phi = -119.74$	$t_\phi = -142.21$	YES
Reactive policy	χ	$b_\chi = -18.55$	$t_\chi = -22.03$	YES

Table 8.3: The main effects for the PSPLIB J30 data set

Technique	Field	Coefficient	T-statistic	Significant?
Intercept		$b_0 = +1819.00$	$t_0 = +173.24$	YES
Initial schedule	τ	$b_\tau = -2.34$	$t_\tau = -0.22$	NO
Resource allocation	υ	$b_\upsilon = -146.82$	$t_\upsilon = -13.98$	YES
Buffer insertion	ϕ	$b_\phi = -1540.69$	$t_\phi = -146.73$	YES
Reactive policy	χ	$b_\chi = -674.99$	$t_\chi = -64.28$	YES

Table 8.4: The main effects for the PSPLIB J120 data set

rameters (e. g. $b_{\tau\upsilon}$) represent the interaction effects when several robustness techniques are used simultaneously. b_0 is the coefficient for the intercept and matches with the average stability cost obtained over the 16 predictive-reactive project scheduling combinations. Y denotes the outcome score of the regression model in terms of average stability cost on the test set.

The remainder of our experimental analysis is organized as follows. First, we will calculate the main effects of the four robustness techniques, which correspond to the fields of our classification scheme, followed by a brief discussion of overfitting. Subsequently, the interaction effects are discussed. Finally, the impact of the amount of variability, the *wp* and the due date setting are analyzed.

8.3.1 The main effects

Tables 8.3 and 8.4 show the parameter estimates for the main effects of the solution robustness techniques on the test set results. All four robustness techniques have a negative parameter value, which denotes that they all improve stability.

Whether these improvements are statistically significant can be investigated by testing the hypothesis $H_0 : b_i = 0$. This is done as follows. We start by calculating the sample variances s_x^2 over all network instances for all 2^4 predictive-reactive scheduling combinations. By assuming that the variance on response values is equal for all 16 combinations, these s_x^2 are all estimates for the variance of the error term of our regression. A better estimate is obtained by pooling the variances into a pooled variance s_p^2:

$$s_p^2 = \frac{\sum_{x=1}^{x=16} s_x^2}{16}. \tag{8.4}$$

On the test set, this results in $s_p^2 = 5450.17$ for PSPLIB J30 and $s_p^2 = 1057820.46$ for PSPLIB J120. Subsequently, in Eq. 8.5 (for J30) and in Eq. 8.6 (for J120) we estimate the variance on any main effect (or interaction effect) as its sample variance:

$$s_{effect}^2 = \frac{s_p^2}{16 \times 480} = 0.71, \tag{8.5}$$

$$s_{effect}^2 = \frac{s_p^2}{16 \times 600} = 110.19. \tag{8.6}$$

Following the central limit theorem, we may assume that the b_i estimates are normally distributed with a variance σ_{effect}^2. The t-statistics (see Tables 8.3 and 8.4) can be calculated as

$$t_i = \frac{b_i}{s_{effect}} \tag{8.7}$$

and follow a Student's with 480×16 and 600×16 degrees of freedom for J30 and J120 respectively. This means that the null hypothesis can be rejected with a 99% probability whenever the t-statistic lies outside the interval $[-2.58, 2.58]$. Consequently, all four robustness techniques have a significant impact on the PSPLIB J30 data set and only initial schedule selection has no significant main effect on the PSPLIB J120 data set.

8.3.1.1 b_τ: initial schedule selection

The main effect of initial schedule selection on robustness is significant on the J30 data set and insignificant on the J120 data set. Because the

experimental analysis on both data sets employs a different algorithm for initial schedule selection, this difference is not completely unexpected. While an exact procedure is used on the J30 data set, we have seen in Chapter 6 that we were restricted to the use of heuristic initial schedule selection procedures for larger networks. Moreover, it will be pointed out later in this chapter that interaction effects cause initial schedule selection to be more important on the J120 data set than its main effect indicates. However, its computational requirement is unarguably heavy. It takes on average just under 1 minute to apply this predictive-reactive scheduling approach to the J30 network instances and just over 1 minute for J120. Remark that applying the metaheuristic of Algorithm 6.4 for initial schedule selection normally generates a buffered schedule S^B. To extract the sole impact of initial schedule selection whenever $\phi = NONE$, this schedule had to be unbuffered first. This was done by applying a serial SGS to the priority list that orders activities in non-decreasing order of their starting times in S^B.

8.3.1.2 b_υ: robust resource allocation

Robust resource allocation has a significant impact on robustness on both data sets, but more importantly, its computational requirement is much smaller than the computational requirement of the initial schedule selection procedures. The computational time required for a simple application of the MABO procedure is on average 0.01 s for the PSPLIB J30 network instances and 0.85 s for the PSPLIB J120 network instances. The MABO procedure has an increased computational requirement (1.50 s) for the procedures of rows 13-16 on PSPLIB J120. This is due to an increased complexity of Step 2.2 in the MABO algorithm (see Algorithm 4.1), because resource allocation is applied to the buffered schedule obtained by the metaheuristic of Algorithm 6.4 instead of an unbuffered initial schedule.

8.3.1.3 b_ϕ: robust buffer insertion

Buffer insertion is shown in Tables 8.3 and 8.4 to have the largest impact of the four factors included in our factorial design. Row 3 in Tables 8.1 and 8.2 shows the results when only STC D is applied to improve robustness. The distinct impact of including buffers covers 94% of the stability gap. Also

less extensive buffer insertion procedures can obtain substantial stability improvements. Proactive-reactive scheduling procedure $|DV|ART|STC|FF\ R|$ only relies on STC to improve solution robustness and obtains average stability costs of 51.20 and 452.32 on the test set for J30 and J120 respectively. These improvements correspond to 93% of the stability gap for both data sets. The computational requirement of STC is on average 0.02 s for J30 network instances and 1.81 s for J120 network instances.

Remark in rows 3 and 4 (or 7 and 8) of Tables 8.1 and 8.2 that the computational time required for buffer insertion with $\phi = STC\ D$ highly depends on the reactive procedure (χ). STC D generates several candidate solutions at each iteration step and evaluates all of them by simulation. In this simulation, the reactive procedure specified by χ has to be applied to every candidate solution. Remember that candidate solutions will be evaluated by EBST rather than SAMP when $\chi = SAMP$.

Row 14 in Tables 8.1 and 8.2 shows the best results obtained among the procedures that do not insert buffers. Extensive initial schedule selection, stable resource allocation and a solution robust reactive procedure together only account for 34% of the stability gap for the J30 data set and 59% for the J120 data set for the current parameter settings. Remark that the results displayed in row 10 are equal to the results of row 14 because the resource flows generated by the resource allocation procedure are not used by the reactive sampling procedure. Figure 8.2 also already established the indispensability of extensive buffer insertion to obtain good results for the parameter settings of this section.

8.3.1.4 b_χ: reactive policy

The reactive policy has the second largest impact on robustness. The difference in the values of t_χ in Tables 8.3 and 8.4 indicates that the use of a more robust reactive policy especially pays off for large-size projects. The sampling procedure obtains 28.9% of the stability gap GP in on average 2.19 s (see row 2) for J30 and 56.8% in 24.97 s for J120. Once again, a far less computationally demanding procedure such as $|DV|ART|NONE|EBST|$ already results in average stability costs of 267.88 (= 23.6% of GP) and 2471.34 (= 51.8% of GP) on the test set for PSPLIB J30 and J120 respec-

tively. This reactive procedure runs on average in 0.01 s on the J30 set and in 0.03 s on the J120 set.

8.3.2 Interaction effects

A similar analysis can be performed for the interaction effects between two or more factors. The parameter estimates and t-statistics of all interaction effects between pairs of factors can be found in Tables 8.5 and 8.6. Higher order interaction effects are shown in Tables 8.7 and 8.8.

Table 8.5: Second order interaction effects for the PSPLIB J30 data set

Interaction	Coefficient	T-statistic	Significant?
τ and υ	$b_{\tau\upsilon} = +0.28$	$t_{\tau\upsilon} = +0.33$	NO
τ and ϕ	$b_{\tau\phi} = +4.42$	$t_{\tau\phi} = +5.25$	YES
τ and χ	$b_{\tau\chi} = +1.46$	$t_{\tau\chi} = +1.73$	NO
υ and ϕ	$b_{\upsilon\phi} = +3.86$	$t_{\upsilon\phi} = +4.58$	YES
υ and χ	$b_{\upsilon\chi} = +5.75$	$t_{\upsilon\chi} = +6.83$	YES
ϕ and χ	$b_{\phi\chi} = +13.81$	$t_{\phi\chi} = +16.40$	YES

Table 8.6: Second order interaction effects for the PSPLIB J120 data set

Interaction	Coefficient	T-statistic	Significant?
τ and υ	$b_{\tau\upsilon} = -31.51$	$t_{\tau\upsilon} = -3.00$	YES
τ and ϕ	$b_{\tau\phi} = +60.57$	$t_{\tau\phi} = +5.77$	YES
τ and χ	$b_{\tau\chi} = -24.32$	$t_{\tau\chi} = -2.32$	NO
υ and ϕ	$b_{\upsilon\phi} = +55.89$	$t_{\upsilon\phi} = +5.32$	YES
υ and χ	$b_{\upsilon\chi} = +146.10$	$t_{\upsilon\chi} = +13.91$	YES
ϕ and χ	$b_{\phi\chi} = +508.31$	$t_{\phi\chi} = +48.41$	YES

On the PSPLIB J120 data set, all but one interaction effects of the second order are statistically significant with a 99% probability. Among them, only the interaction between initial schedule selection and robust resource allocation ($b_{\tau\upsilon}$) is negatively correlated and has thus a positive impact on the minimization of our stability cost function. In absolute value, $b_{\tau\upsilon}$ is the smallest coefficient among the significant interaction effects. On PSPLIB

J30 (see Table 8.5), it is even not statistically significant with a 99% probability.

The most significant second order interaction effect arises between buffer insertion (ϕ) and reactive scheduling (χ). This interaction effect implies that a robust reactive policy will be less performing whenever buffers have already been inserted. This can be simply verified in rows 1-4 of Tables 8.1 and 8.2 by comparing the improvements made by applying a robust reactive policy when buffers have been inserted with the improvements made by applying a robust reactive policy when no buffers have been inserted. Buffer insertion by itself already performs as well that only few improvements can be added by additional robustness techniques. However, because $b_\chi + b_{\phi\chi} < 0$, the overall impact of reactive scheduling on our stability cost function remains negative. Reactive scheduling still pays off after buffer insertion.

Table 8.7: Higher order interaction effects for the PSPLIB J30 data set

Interaction	Coefficient	T-statistic	Significant?
τ, υ and ϕ	$b_{\tau\upsilon\phi} = -0.13$	$t_{\tau\upsilon} =$	NO
τ, υ and χ	$b_{\tau\upsilon\chi} = -0.27$	$t_{\tau\phi} =$	NO
τ, ϕ and χ	$b_{\tau\phi\chi} = -0.91$	$t_{\tau\phi\chi} =$	NO
υ, ϕ and χ	$b_{\upsilon\phi\chi} = -4.08$	$t_{\upsilon\phi\chi} =$	YES
τ, υ, ϕ and χ	$b_{\tau\upsilon\phi\chi} = +0.10$	$t_{\tau\upsilon\phi\chi} =$	NO

Table 8.8: Higher order interaction effects for the PSPLIB J120 data set

Interaction	Coefficient	T-statistic	Significant?
τ, υ and ϕ	$b_{\tau\upsilon\phi} = -27.38$	$t_{\tau\upsilon} = -2.61$	NO
τ, υ and χ	$b_{\tau\upsilon\chi} = +31.57$	$t_{\tau\phi} = +3.00$	YES
τ, ϕ and χ	$b_{\tau\phi\chi} = -33.63$	$t_{\tau\phi\chi} = -3.20$	YES
υ, ϕ and χ	$b_{\upsilon\phi\chi} = -56.61$	$t_{\upsilon\phi\chi} = -5.39$	YES
τ, υ, ϕ and χ	$b_{\tau\upsilon\phi\chi} = +27.44$	$t_{\tau\upsilon\phi\chi} = +2.61$	NO

Tables 8.5 and 8.6 also show that there is a very significant interaction between robust resource allocation and reactive scheduling ($t_{\upsilon\chi}$). We even remark that $b_\upsilon + b_{\upsilon\chi} \approx 0$, which denotes that the robustness introduced by

a more robust resource allocation is completely neutralized when a robust schedule repair mechanism is used upon schedule breakage. This might not be a surprise, because the main interest of robust resource allocation is to construct a resource flow network that can be preserved as a reactive policy ($\upsilon = FF\ R$). When a priority rule ($\upsilon = EBST$) or the sampling procedure ($\upsilon = SAMP$) are used for rescheduling, the impact of a more robust resource allocation is limited to its use in buffer insertion algorithms.

A related third order effect comes here into prominence. The significant YES-indication for $b_{\upsilon\phi\chi}$ in Tables 8.7 and 8.8 states that only when buffers are inserted in the baseline schedule, robust resource allocation improves the stability cost that will be obtained when applying the sampling procedure for rescheduling. If no buffers are inserted, generating a resource flow network becomes completely redundant.

On the J120 data set, we remark a second significant third order effect ($b_{\tau\upsilon\chi}$) that includes resource allocation and reactive scheduling. This interaction becomes perceptible by pointing out the large difference on obtained stability cost between rows 11 and 15 in Table 8.2. Because the metaheuristic that is used for initial schedule selection ($\tau = BEST$) on the J120 data set delivers a buffered input schedule for the resource allocation procedure, the resource allocation problem becomes more complex and the potential gains of a robust resource allocation procedure more pronounced. Of course, this requires special attention for solving the resource allocation problem when $\chi = FF\ R$.

We illustrate the significance of $b_{\tau\upsilon\chi}$ on a small example. Consider the partial schedule of Figure 8.3 with a buffer inserted between activities 2 and 3, which may be due to a high w_3. A non-robust resource allocation procedure may decide to add a resource flow between activities 1 and 3, instead of between activities 2 and 3. Keeping this resource flow between activities 1 and 3 fixed, may substantially harm the solution robustness of the schedule. On the J30 data set, the initial schedule is unbuffered and this type of interaction cannot arise.

There also exists a significant interaction effect between initial schedule selection (τ) and buffer insertion (ϕ). The improvement heuristic for buffer insertion is able to transform initial solutions with inferior robustness into

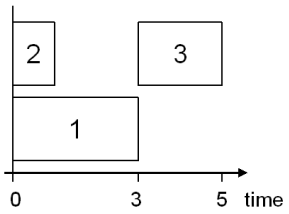


Figure 8.3: Potential interaction effect between τ, υ and χ

stable buffered baseline schedules. While it still holds on PSPLIB J30 that $b_\tau + b_{\tau\phi} < 0$, this inequality becomes $b_\tau + b_{\tau\phi} \gg 0$ for the J120 data set. The larger value of $t_{\tau\phi}$ on PSPLIB J120 can be explained by a basic understanding of the internal logic of the applied metaheuristic for initial schedule selection. The initial schedule found is guaranteed to be active, but might have a larger makespan than the initial schedule selected by $\tau = DV$. As has been shown in Figures 6.1 and 6.2, buffer insertion may invert the impact of such near-optimal initial schedules on solution robustness. We may conclude that the relatively small impact of the computationally demanding initial schedule selection procedures only may assist in a valuable contribution for solution robustness when no buffers are inserted in this schedule.

Roughly the same reasoning can be given to interpret the significant value of $b_{\upsilon\phi}$. Resource allocation has unarguably its merits for solution robust scheduling (t_υ), but the significant interaction effect between υ and ϕ reveals that extensive buffer insertion procedures such as STC D, which start from a given resource allocation, are able to compensate for the weaknesses in the resource flow network. However, as already stated above, applying robust resource allocation with the MABO heuristic has the undeniable advantage that it is computationally far less heavy than our initial schedule selection procedures.

8.3.3 Overfitting

All tables in this chapter show results on both a training set and a test set of simulated disruptions scenarios. Some of the procedures applied in this experiment rely on simulation to evaluate multiple candidate solutions and select the best one among them. Using the same set of disruptions to

evaluate candidates and to evaluate the objective function of the predictive-reactive scheduling approach, could lead to overfitting. This means that the procedure at hand fits its selection procedure too much to the generated disruptions. The best schedule for the simulated disruptions will not necessarily be the best schedule for the actual disruptions during project execution, even if we assume that they are drawn from the same density functions.

Overfitting is detected by examining the results on both the training and test set of execution scenarios. The training set is used to evaluate several candidate solutions. In particular the buffer insertion algorithm STC D is subject to overfitting, but also the initial schedule selection procedure BEST and the MABO algorithm for solution robust resource allocation may suffer from overfitting. The reactive sampling approach also compares multiple candidates, but this is done for one scenario rather than for averages over a set of scenarios. Sampling does not suffer from overfitting.

Table 8.9: Overfitting on PSPLIB J120

τ	υ	ϕ	χ	Training	Test	Time
DV	ART	STC	FF R	450.85	452.35	3.79
DV	ART	STC D	FF R	356.10	391.33	12.90

Overfitting can be easily illustrated when comparing the results in both rows of Table 8.9. The difference on stability cost between STC and STC D has partially vanished on the test set results, because the improvement phase is subject to overfitting. The above factorial designs are all applied to the test results to limit the impact of overfitting on our conclusions.

8.3.4 The impact of activity duration variability

The results in Tables 8.1 and 8.2 and the conclusions made so far only hold for the given parameter settings. In this section, we will examine the impact of reduced variability in the activity durations on solution robustness. We will concentrate our efforts on the PSPLIB J120 data set. Results are shown in Table 8.10.

Reduced activity duration variability obviously decreases the stability

Table 8.10: Performance values on PSPLIB J120 when variability is low

Row Nr.	τ	υ	ϕ	χ	Training	Test	Time
1	DV	ART	NONE	FF R	3513.95	3522.70	1.98
2	DV	ART	NONE	SAMP	1689.52	1692.69	20.07
3	DV	ART	STC D	FF R	81.11	93.71	11.31
4	DV	ART	STC D	SAMP	40.81	43.03	95.40
5	DV	MABO	NONE	FF R	2929.66	2952.82	2.82
6	DV	MABO	NONE	SAMP	1689.52	1692.69	20.91
7	DV	MABO	STC D	FF R	56.10	65.36	12.11
8	DV	MABO	STC D	SAMP	27.32	32.59	100.60
9	BEST	ART	NONE	FF R	3420.72	3425.45	64.53
10	BEST	ART	NONE	SAMP	1607.60	1610.32	82.09
11	BEST	ART	STC D	FF R	264.22	290.36	73.42
12	BEST	ART	STC D	SAMP	29.48	34.84	146.16
13	BEST	MABO	NONE	FF R	2810.95	2826.57	65.86
14	BEST	MABO	NONE	SAMP	1607.60	1610.32	83.43
15	BEST	MABO	STC D	FF R	55.16	66.76	73.23
16	BEST	MABO	STC D	SAMP	27.97	33.72	162.81

cost because activities are more likely to start on their planned starting time. We see for example that the stability cost in row 1 is reduced with almost 30% compared to Table 8.2. However, when buffers are included (e.g. rows 3, 8, 12 and 16), the obtained reduction in stability cost is even far more pronounced. By only including buffers ($\phi = STC\ D$) in the baseline schedule, we observe that we have a reduction of 76% compared to the same strategy in the high variable environment and that 98% of the stability gap can already be obtained. Moreover, it has been shown in the paper by Van de Vonder et al. (2005b) that especially in environments with low variability, quality robustness suffers little from including buffers in the baseline schedule. Contrary to common beliefs, this results in the fact that solution robust scheduling (by inserting buffers) can be regarded as even more beneficial in environments with low variability.

Initial schedule selection is again only beneficial if no advanced buffer

insertion procedure is used. In Table 8.10 we even obtain slightly better results in row 8 than in row 16. Solution robust resource allocation ($\upsilon = MABO$) has a similar impact for low and high variability parameter settings.

8.3.5 The impact of the project due date

Table 8.11: Performance values on PSPLIB J120 when the due date is tight

Row Nr.	τ	υ	ϕ	χ	Training	Test	Time
1	DV	ART	NONE	FF R	5452.25	5465.33	1.98
2	DV	ART	NONE	SAMP	2400.82	2414.13	24.17
3	DV	ART	STC D	FF R	2397.83	2454.63	9.32
4	DV	ART	STC D	SAMP	849.81	871.40	61.68
5	DV	MABO	NONE	FF R	4535.37	4571.91	2.85
6	DV	MABO	NONE	SAMP	2400.82	2414.13	25.04
7	DV	MABO	STC D	FF R	1796.95	1863.52	10.27
8	DV	MABO	STC D	SAMP	833.15	857.16	69.05
9	BEST	ART	NONE	FF R	5299.23	5321.34	39.92
10	BEST	ART	NONE	SAMP	2108.34	2123.74	62.18
11	BEST	ART	STC D	FF R	3067.96	3118.31	44.97
12	BEST	ART	STC D	SAMP	810.12	836.55	93.21
13	BEST	MABO	NONE	FF R	4341.1	4383.58	41.18
14	BEST	MABO	NONE	SAMP	2108.34	2123.74	63.45
15	BEST	MABO	STC D	FF R	1763.40	1842.83	47.20
16	BEST	MABO	STC D	SAMP	793.32	823.05	106.47

Most project managers assume that buffer management would not be an appropriate approach for their project because their due date is tight. In Chapter 3 we already countered this statement by emphasizing the importance of the weight of the dummy end activity to protect the project due date in solution robust procedures. In this section, we will reinvestigate the impact of solution robust project scheduling when the project has a tight project due date. Results are shown in Table 8.11.

The first observation that we can make is that the best obtainable stability cost of row 16 is much higher than before. Solution robust scheduling

now becomes challenging and including buffers is no longer a guarantee for satisfying results. We see in row 7 of Table 8.11 that buffer insertion only accounts for 64% of the stability gap. The tight due date makes the last activity very critical and including buffers can for this reason only be done to a certain extent. Also, proactive scheduling by initial schedule selection becomes challenging because of the tight due date. Remember that initial scheduling obtained a part of its improvement by opting for a near-optimal initial schedule with more inherent slack. Such schedules will typically induce a high stability cost of the last activity when the project due date is tight.

The effectiveness of solution robust resource allocation increases slightly, but especially reactive scheduling becomes critical in this setting. Applying a factorial design on the results of Table 8.11 even reveals that b_χ slightly surpasses b_ϕ to give the reactive scheduling policy the most significant main effect. A due date that can be considered as tight for the uncertainty included in a certain project impels advanced solution robust reactive procedures.

8.3.6 The impact of the weighting parameter

The weighting parameter wp was uncovered as the driving force behind the trade-off between makespan and stability in Chapter 3. In this section, we will investigate the impact of wp on the magnitude of the impact of the different predictive-reactive project scheduling procedures.

The wp has an almost negligible impact on the results obtained by the predictive-reactive procedures that have no buffers included (rows 1-2, 5-6, 9-10 and 13-14). These procedures aim at quality robustness and will typically obtain a high timely project completion probability (TPCP). The weight w_n that is accorded for each time unit of exceeding the project due date is thus less important for these procedures.

Buffer insertion procedures, however, highly rely on the weighting parameter to decide how to divide safety time between intermediate buffers and project buffers. A smaller wp will allow to increase the buffer sizes of intermediate activities, resulting in lower intermediate stability losses. We

Table 8.12: Performance values on PSPLIB J120 when $wp = 5$

Row Nr.	τ	υ	ϕ	χ	Training	Test	Time
1	DV	ART	NONE	FF R	4989.85	5003.25	1.98
2	DV	ART	NONE	SAMP	2216.92	2229.10	24.58
3	DV	ART	STC D	FF R	304.78	334.65	13.45
4	DV	ART	STC D	SAMP	91.33	101.33	106.32
5	DV	MABO	NONE	FF R	4182.07	4214.76	2.84
6	DV	MABO	NONE	SAMP	2216.92	2229.10	25.44
7	DV	MABO	STC D	FF R	206.99	231.9	13.84
8	DV	MABO	STC D	SAMP	88.75	98.78	114.91
9	BEST	ART	NONE	FF R	4869.62	4888.50	57.52
10	BEST	ART	NONE	SAMP	1756.34	1765.83	79.11
11	BEST	ART	STC D	FF R	697.99	740.49	68.15
12	BEST	ART	STC D	SAMP	91.43	102.24	150.75
13	BEST	MABO	NONE	FF R	4021.63	4057.39	58.99
14	BEST	MABO	NONE	SAMP	1756.34	1765.83	80.58
15	BEST	MABO	STC D	FF R	198.56	228.91	68.13
16	BEST	MABO	STC D	SAMP	88.20	99.72	169.88

see in Table 8.12 that all procedures with $\phi \neq NONE$ obtain a stability improvement of close to 10% compared to their results in Table 8.2. The proactive-reactive procedure of row 16 could for example achieve a 9.83% stability improvement: from 110.60 in Table 8.2 to 99.72 in Table 8.12. This improvement has been obtained by increasing the intermediate safety and allowing a slightly higher number of project executions that exceed the project due date. If we would have used the same schedules generated by the improvement heuristic for the results of Table 8.2 as baseline schedules, only $w_n|s_n^T - \delta_n|$ would change in the objective function of Eq. 2.3. The objective function value would drop to 100.19 on the test set. Hence, a lower wp does not only result in a lower stability cost because exceeding the project due date is punished less, but also because more safety can be included in the baseline schedule.

8.4 Conclusions

The overall objective of this chapter was to evaluate the performance of various predictive-reactive project scheduling classes under the stability objective function.

The factorial designs in this chapter showed that buffer insertion is the most effective technique to improve the solution robustness of a project. However, when exploiting a composite quality robustness / solution robustness objective function such as the objective function introduced in Eq. 2.12, buffer insertion procedures may be subject to low quality robustness whenever the variability is high and the due date is tight. Although we will argue that such a lack of quality robustness stems from an underestimation of wp, we will discuss in the next chapter that buffer insertion might suffer from acceptance issues in many real-life environments because of this seeming lack of quality robustness.

For such projects, techniques such as initial schedule selection and solution robust resource allocation are valuable alternatives to improve the solution robustness of the baseline schedule, especially when computationally efficient procedures such as MABO can be applied. There are few practical objections against the use of such solution robust resource allocation procedures.

For initial schedule selection, the main issue is that multiple candidate solutions need to be evaluated and that this should preferably be done after buffer insertion. The initial schedule selection procedures proposed in Chapter 6 are all rather time consuming for realistic project networks. Hence, their small positive impact on stability is harder to justify than the impact of resource allocation. The two-staged approach, which has been advocated to be interesting for practical reasons in Section 2.1.2.2, also shows to be defendable based on performance. The integrated proactive scheduling problem that includes scheduling, resource allocation and buffer insertion is very complex and still needs further research.

The reactive scheduling procedures of Chapter 7 show to have a substantial impact on solution robustness. When the due date can be considered as tight, given the amount of variability in the activity durations, it was even shown that a robust reactive policy was equally effective as buffer insertion.

Even computationally light procedures such as the robust SGSs of Chapter 7 allow to considerably improve stability and can be used in simulation approaches where multiple candidate schedules require evaluation. For the unique actual project execution, more advanced reactive procedures can be applied to further improve the schedule stability.

Solution robust predictive-reactive procedures have also shown to posses a large capability to adapt to the given parameter settings. Even when variability is rather low or the project due date is very tight and critical, considerable improvements are possible.

Chapter 9

A real-life application

The wide gap between theory and practice is a huge problem in project scheduling. Researchers develop procedures that are often too restricted for real-life projects. On the other hand, most project managers are simply not aware of the existing procedures that could really help them improving the quality of their project. In this chapter, we would like to address issues concerning the uncertain environment in which real-life projects take place. We had the opportunity to participate in the project *Risk Management in the Construction Industry*, a large scaled collaboration between the Belgian Building Research Institute (BBRI) and our university that was supported by several important Belgian construction firms[1].

We aim to apply some of the algorithms proposed in the previous chapters of this thesis to a real-life project. We hope that the conclusions will help project managers to understand the need for extensive project scheduling and help researchers to concentrate their future research on real-life based problems.

The outline of this chapter is as follows. We start by defining risk management, which is inevitably related to robust project scheduling. Afterwards, in Section 9.2 a framework is provided that gives an in-depth analysis of the risk management approach that we propose based on real-life experience and theoretical insights gained in the previous chapters of this work. This framework will then be applied to an example project in Section

[1]Project QP6-353901-IWT-Risicomanagement in de bouw

9.3. Afterwards, Section 9.4 offers a discourse concerning communication and acceptance of robust project planning. The chapter will be concluded by summarizing the main conclusions and the main fallacies of our approach.

9.1 Risk management

Robust scheduling is a crucial part of risk management for project planning, but the subject of risk management implies much more than only scheduling. Figure 9.1 represents risk management as an iterative process. In the remainder of this section, the phases of this process (see also Chapter 11 in PMBOK® Guide, 1996[2]) will be described in some detail.

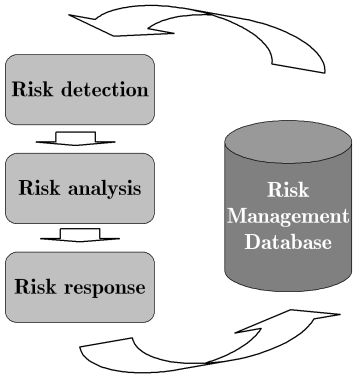


Figure 9.1: Risk management as an iterative process

9.1.1 Risk detection

What are the risks that the current project is subject to? This seemingly easy question is shown to be difficult to answer appropriately by most practitioners. Existing checklists (e.g. van Well-Stam et al. (2003)) of possible risks for several tasks might substantially help in this matter. One of the main aims in the Risk Management in the Construction Industry project is to compose such lists (see Section 9.1.4). Risk identification is beyond the scope of this thesis.

[2]The Guide to the Project Management Body of Knowledge is published by the Project Management Institute (PMI) and proposes a standard for project managent

9.1.2 Risk analysis

While gathering information about risks is already regarded as a difficult task, analyzing these risks causes even far more problems for most real-life projects. The basic requirement for risk analysis to be accepted by project management is that it has to be straightforward and fast. We cannot expect project practitioners to have either advanced mathematical knowledge or redundant time. On the other hand, the output of risk management is completely relying on the quality of this risk analysis. This results in the necessity of an understandable and user-friendly graphical user interface (GUI) to collect the required information within an acceptable period of time. We feel that a quantitative approach to risk analysis strongly improves the effectiveness of risk management. Fuzzy concepts as "rather often" or "almost certainly" are too vague to effectively model uncertainty and will result in misunderstandings between managers, planners, consultants and workforce.

9.1.3 Risk response

Once the risks have been detected and analyzed, the organization has to decide how to address these risks. Based on Section 11.3 of the PMBOK® (1996), we distinguish several ways of tackling risks:

- **Risk reduction**
  Risk detection and prioritization often uncover hidden information such that project managers can reduce risks by changing techniques or execution modes (f.i. by crashing activity durations) during project execution.

- **Risk transfer**
  Here, it is made sure that the possible negative consequences of the risk lie outside the organization, f.i. by subcontracting or risk insurance.

- **Risk treatment**
  The organization is aware of the risk and knows exactly how to react when the risk event occurs. This is the most reactive technique and

might be the preferable technique for risk events that have a very small probability to occur, but a high impact.

- **Risk acceptance**
  Some (mostly unsignificant) risks should just be accepted because any other risk response measure would be more costly to implement than the possible negative outcome of the risk itself.

- **Risk anticipation**
  Recognize the risks as an integral part of the project and anticipate its consequences by planning them as events that will happen with a certain probability. This is where robust project scheduling comes into play.

A detailed examination of risk reduction, transfer and treatment lies outside the scope of this chapter. We assume that the organization has already allotted sufficient attention to these techniques and that only the residual risks have to be taken into account in the project scheduling phase, which remains the central point of focus in this chapter. Risk acceptance can be considered as an extreme case of risk anticipation in which the preferable reaction to anticipate the risk is doing nothing.

9.1.4 The risk management database

The risk management database is in theory the heart of risk management. Any project is by definition unique, but mostly neither its activities nor the related risks are. A risk management database should contain historical data about all risks that might affect the activities in a project and the potential measures (risk responses) to combat these risks. The actual disruptions that occur during project execution and their consequences should also be stored. Risk management done for one project (or activity) can facilitate the work for upcoming similar projects (or activities). In practice, however, not many real-life projects can rely on a reliable database with relevant risks.

9.2 A framework for risk management

In this section, a formal framework for risk management will be introduced. Risk management is present in every single phase of the project life cycle of Figure 2.6. Our proposed framework will also fit into the subsequent phases of the life cycle. Rather than giving an overview of all decisions that have to be made during this process, we focus on some recommendations that are important for risk management.

9.2.1 Project initiation phase and project definition phase

Before the start of the project planning phase, the organization must be clear about the whereabouts of the project. The preferable planning and scheduling techniques are dependent on which objectives have been set. It is pointless to use commercial project scheduling software packages without being aware of what this software is trying to optimize.

In this thesis, we focused on projects that appreciate a stable execution with reduced nervousness. We must note that also for organizations that focus on time-criticality, schedule robustness might be an important secondary objective function.

In this phase, strategic and tactical project constraints such as the due date and the budget should also be identified.

9.2.2 Project planning phase

Once the project has been properly defined and the importance of risk management for the current project has been uncovered, the project planning phase can be started. Gathering and processing information is crucial to this phase. In this section we will describe what information is required and how we propose to gather it. Although this section is based on interviews between a consultant and a single project manager, we would strongly advise to collect multiple opinions for large-scaled projects, if possible. Historical data can also help to improve estimations. Aggregated estimates will be less biased and better accepted by the workforce.

9.2.2.1 Identifying the activities

The focus during the activity identification will shift compared to standard project scheduling. In a first interview between the planner (consultant) and the project manager the required data are collected.

Estimating activity durations is a compulsory first task for project managers that apply our risk management framework. However, the interviewer should stress that aggressive duration estimates d_j^* are required, in which no safety should be included whatsoever[3]. The resulting aggressive duration might very well be utopian. The idea behind these aggressive estimates mirrors the basic idea of risk management, being that we aim to make project managers conscious about uncertainty. To some extent, uncertainty is always included in a project schedule, but this inclusion rarely happens consciously. Managing such instinctive uncertainty is impractical.

Because most projects contain too many activities to be analyzed effectively, activities will be grouped into a limited amount of *activity groups*, containing activities with similar risk structure. Remark that activity grouping should not be confused with hierarchical aggregating activities in more generalized work packages (cf. work breakdown structure). In a work breakdown structure we could for example aggregate the activities "painting the bedroom ceiling" and "tiling the bedroom" into the aggregated activity "bedroom finalization", while we are interested in grouping all painting activities in a single activity group, regardless of where and when the individual group activities are executed. The idea is that the grouped painting activities will suffer from similar potential risks because similar techniques, subcontractors, etc. are applied.

Grouping the activities that share common risks into activity groups, simplifies the subsequent risk process, which can now be performed at the activity group level rather than at the individual activity level.

For each activity group, the project manager must also specify the

[3]It should be remarked that this definition of the term aggressive activity duration does not correspond with its use in the CC/BM methodology of Chapter 3, where aggressive activity durations denote average or median activity durations. The aggressive estimates in this chapter correspond to the (unrealistic) best case duration rather than the 50% confidence duration.

activity weight w_j to be assigned to each activity in the group. This weight represents a marginal disruption cost of starting the activity during project execution earlier or later than planned in the baseline schedule. The weights reflect the scheduling flexibility of the activities in the groups and will be used by the robust project scheduling procedures described in Section 2.2. A small activity weight reflects high scheduling flexibility or low instability cost: it does not 'cost' that much if the actually realized activity starting time during schedule execution differs from the planned starting time in the baseline schedule. A heavy weight reflects low scheduling flexibility: deviations between actual and planned starting times are deemed very costly for the organization (e.g. high penalties that are incurred when individual milestones or the project due date are not met).

Information about resource requirements, precedence constraints, scheduling constraints, etc. should also be gathered in this phase.

9.2.2.2 Constructing the project network

Many project scheduling procedures in literature start from a graphical activity-on-node network representation ($G(N, A)$) of the project in which N defines the set of activities in the project and A specifies the existing precedence relationships between the activities. Such a valid project network may not contain activities without successors, except the dummy end activity. Circular references are only allowed when generalized precedence relations (Elmaghraby & Kamburowski 1992) occur.

Our experience indicates that project managers tend to generate precedence relations that already reflect implicit scheduling decisions, rather than pure technologically based precedence constraints. For example, two activities a and b may be assigned a finish-start, zero-lag precedence relation by project management, not because technological conditions impose such a relationship, but because both activities require the same resource that is available in a single unit, or because company tradition calls for executing activity a first. The result is that a precedence and resource feasible schedule is generated for a project that will suffer from unnecessary precedence constraints, with an unjustified reduction in scheduling flexibility, and an unnecessary propagation of scheduling conflicts caused by scheduling dis-

ruptions that may occur during project execution.

9.2.2.3 Detection of risks and opportunities

For each identified activity group, project management has to determine the relevant risk factors. A limited list of potential risk factors can be extracted from the risk management database or new risk factors may de determined for the project under consideration.

The potential impact of the risk on the project depends on the risk-specific characteristics. Some of the possible risk impacts may be:

- Activity duration increase (in time units)
- Productivity decrease (in percentage)
- Delay of activity start time
- Increase of costs
- Increase of resource requirements

The difference between activity duration increase and productivity decrease lies in the fact that in the latter case, the impact depends on the activity duration itself. For example, the impact of bad weather conditions can be expressed as an extra percentage of required time.

Also the rate of occurrence can be either dependent or independent of the activity duration. Risks that affect the activity start time mostly have both a probability and an impact that are unrelated to d_j. In Section 9.2.2.5, it will be described how these differently characterized risks will be coped with.

9.2.2.4 A scenario-based technique for risk analysis

Simple quantification of risks is a huge challenge. Our approach is somewhat similar to the *simple scenario approach* of Chapman & Ward (2003). Similar to theirs, our approach expects project management to provide the probability of occurrence and the impact of the risk factors on

the activities of an activity group under a best case scenario and a worst case scenario.

The graphical user interface (GUI) prompts the project manager to answer a list of scenario based questions. The questions need to be clear, non-misleading and intuitive. Questions such as "what are the probability and impact of this event?" do not reflect the mangers' reasoning. An example risk quantification session between the expert (E) and the project manager (M) could look as follows.

E (1): Imagine a worst case scenario. How long will the affected activity duration be extended by this risk in the worst case scenario?
M (2): b days
E (3): Based on your past experiences, what is the frequency that such a worst case scenario has appeared in a similar project?
M (4): $\zeta(b)$
E (5): Assume now that the risk occurs, but the prolongation of the affected activity duration can be limited as much as possible. What will this best case prolongation be?
M (6): a days
E (7): Based on your past experiences, what is the frequency that such a best case scenario has appeared in a similar project?
M (8): $\zeta(a)$
E (9): Finally, what is the overall frequency that such a risk has appeared in a similar project?
M (10): q

The frequencies $\zeta(a), \zeta(b)$ and q can be larger than one if the project manager expects several occurrences of the risk during the project. The data are entered in the GUI after validation. A simple validation test states that $q \geq \zeta(a)+\zeta(b)$. Otherwise, revision by the project manager is required.

From the two extreme case point estimates, (a and b) a triangular probability density function $f(\mathbf{x})$ and its cumulative distribution function $F(\mathbf{x})$ for the impact $\mathbf{x}$ are generated. A triangular distribution is completely defined by three parameters, being the lower limit, the mode and the upper limit.

The first step to generate $f(\mathbf{x})$ is the determination of c, the most likely

estimate for the impact. Our experience shows that project managers often struggle to estimate c because of the fuzziness inherent to the "most likely" concept. We propose to calculate c instead. For the time being, we assume that the best case scenario a and the worst case scenario b are the lower and upper limit of $f(\mathbf{x})$ respectively. We find that

$$c = \frac{a\zeta(a) + b\zeta(b)}{\zeta(a) + \zeta(b)} \tag{9.1}$$

The reasoning behind this formula starts from the idea that project managers never think in terms of point estimates $f(x)$ for a continuous distribution. We do, however, assume that the fraction $\zeta(a)/\zeta(b)$ is a correct estimate of $P(a)/P(b)$ in which $P(a)$ and $P(b)$ (see Eq. 9.2 and 9.3) must be interpreted by the expert as being the discrete probabilities that if the risk occurs, its impact lies within a certain scenario interval with width ϵ. ϵ reflects to what extent the project manager reasons in terms of scenarios and is highly dependent on the individual. $\zeta(a)/\zeta(b)$ is thus regarded as the ratio that the best case scenario is thought to be more likely to occur than the worst case scenario.

$$P(a) = \int_{x=a}^{x=a+\epsilon} f(x)dx = F(a+\epsilon) \tag{9.2}$$

$$P(b) = \int_{x=b-\epsilon}^{x=b} f(x)dx = 1 - F(b-\epsilon) \tag{9.3}$$

From 9.2 and 9.3 we may conclude that

$$\frac{\zeta(a)}{\zeta(b)} = \frac{P(a)}{P(b)} = \frac{F(a+\epsilon)}{1 - F(b-\epsilon)} \tag{9.4}$$

Next, the distribution function $F(x)$ of a triangular distribution is known and defined by:

$$F(x) = \begin{cases} \frac{(x-a)^2}{(b-a)(c-a)} & \text{for } a \leq x \leq c \\ 1 - \frac{(b-x)^2}{(b-a)(b-c)} & \text{for } c < x \leq b \end{cases} \tag{9.5}$$

Substituting 9.5 in 9.4 gives

$$\frac{\zeta(a)}{\zeta(b)} = \frac{\frac{\epsilon^2}{(b-a)(c-a)}}{\frac{\epsilon^2}{(b-a)(b-c)}} = \frac{b-c}{c-a} \tag{9.6}$$

Because $b - c = (b - a) - (c - a)$ we now find that

$$c = a + \frac{b - a}{1 + \frac{\zeta(a)}{\zeta(b)}} \tag{9.7}$$

Reformulating equation 9.7 results in equation 9.1. This formula is independent of the actual values of ϵ, $\zeta(a)$ and $\zeta(b)$. Figure 9.2 shows a triangular density function $f(\mathbf{x})$ constructed based on the parameters a, b and c. Note that $\zeta(a)/\zeta(b)$ equals 3 in this figure. The cumulative distribution function $F(\mathbf{x})$ of $f(\mathbf{x})$ is shown in Figure 9.3.

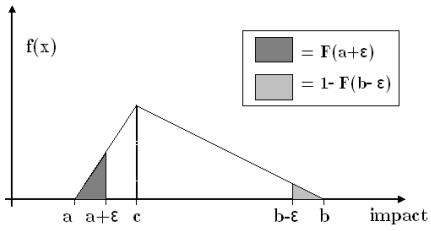


Figure 9.2: A triangular probability density function $f(x)$

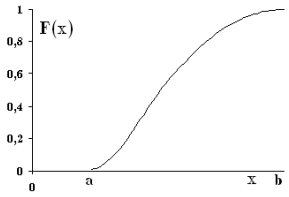


Figure 9.3: A cumulative distribution function $F(x)$

9.2.2.5 Deriving individual activity risk profiles

The probability density functions $f(\mathbf{x})$ of the impacts of all detected risks, derived in the previous section at the level of the activity groups, have to be projected on the individual project activities. This projection is the key to the reusability of the risk assessments at the level of the activity groups. Distributions for project activities can be calculated by using the

characteristics of the risk impact densities and probability, risk assessments and/or risk data coming from project records.

In the previous section we described how the GUI was used to prompt the project manager to specify the average occurrence rate q of any risk in activity group AG. To obtain the probability density function $f(\mathbf{d}_j)$, we rather need the occurrence rate of the risk at the level of a single activity j.

We assume that the number (k) of risk occurrences is Poisson distributed (see Eq. 9.8) and that the rate λ_j at which the risk occurs for activity j can be calculated by Eq. 9.9.

$$f(k;\lambda_j) = \frac{e^{-\lambda_j}\lambda_j^k}{k!} \tag{9.8}$$

$$\lambda_j = \frac{v_j}{\sum_{\forall i \in AG} v_i} q \tag{9.9}$$

The weights v_i depend on the characteristics of the risk. If the risk occurrence rate is independent of d_i (e. g. license delay), we set $v_i = u_i$ where u_i is a binary variable that equals 1 if the risk affects activity i and equals 0 otherwise. When the occurrence depends on d_i, we set $v_i = d_i^* \times u_i$ where d_i^* is the aggressive duration estimate of activity i.

Simulation can then be used to obtain a probability density function $f(\mathbf{d}_j)$ for the activity duration $\mathbf{d}_j$. We start by simulating the number of occurrences k which follows the distribution of Eq. 9.8. Then, for each occurrence l ($l = 1, \ldots, k$), the inverse distribution function $\mathbf{x} = F^{-1}(\mathbf{y})$ can be used to generate random values x_l for the impact with $\mathbf{y}$ being drawn from a uniform distribution between 0 and 1.

Subsequently, the simulated duration d_j is calculated as a function of the aggressive activity duration d_j^* and the simulated impacts x_l for each l. How this is done depends on the characteristics of the impact of the risk. For example, for risk factors that may lead to an activity duration increase - our major concern in this chapter - we obtain the expected activity duration as:

$$d_j = d_j^* + \sum_{l=1}^{k} x_l. \tag{9.10}$$

A sufficient number of simulation runs allows for the generation of the distribution functions $f(\mathbf{d}_j)$. These simulated distribution functions are

then fit into known distribution functions f^*. A triangular distribution, for example, is completely characterized by its lower limit a_j, its upper limit b_j and its mode c_j. a_j en b_j are directly distilled from the simulated f and c_j is calculated as:

$$c_j = 3 \times E(\mathbf{d}_j) - a_j - b_j. \tag{9.11}$$

Visibility for project managers is the main advantage of this approach. a_j, b_j and c_j refer to an optimistic, a pessimistic and a most likely estimate of $\mathbf{d}_j$.

9.2.2.6 Validating the estimates

The estimates a_j, b_j and c_j are the result of subjective parameter estimates and need to be handled with care. The project manager who detected and analyzed the risks should validate the resulting parameters by consulting the historical data in the risk management database and/or gathering expert opinion.

It should be clear that, when the risk analysis procedure described above is deemed too extensive, the three point estimates of $\mathbf{d}_j$ may be directly determined on the basis of past experience or historical data.

Applying a sensitivity analysis of the risk parameter estimates might provide additional insight in the robustness of the estimates. A project manager could overestimate the worst case impact of a risk to make a statement. Showing him the impact of this overestimation, could change his mind.

9.2.2.7 Risk responses

As we stated above, the project manager has to ponder on how to respond to risks. The possible risk reducing measures should all be added to the risk management database. Some examples are outsourcing a certain activity or activity group, getting an insurance for a financially huge risk, performing an alternative technique which does not contain the risk, adding more workforce, etc. Thanks to the extensive risk analysis in the previous section, risk prioritization helps to decide which risks are the most important to address.

Robust project scheduling can be regarded as a risk response measure, but deserves special attention in the next section.

9.2.3 Project scheduling phase

Most decision making in project scheduling is situated in this phase of the project life cycle. The aim of the scheduling phase is to obtain a workable baseline schedule for the project and to investigate how risk management can influence our decisions.

Risk management makes project planning and scheduling an iterative process (see Figure 9.1). Many of the above risk responding measures have an impact on the project planning phase. Applying a different technique to perform an activity might for example reduce or eliminate a certain risk. A robust project plan (risk anticipation) may change the impact of an activity duration increase. The residual risks need to be re-detected and a new cycle of risk management starts.

Once residual risks have been obtained and the iterative process has been stabilized, a workable project schedule will be generated. Generating a robust project plan aims to reduce the schedule nervousness. Typically, unrealistic project plans are built that are closely followed in the early phase of project execution without looking at uncertainty and are then inevitably abandoned in later phases, resulting in project anarchy. Dealing with risks in a well-organized way is crucial in an uncertain project environment. The later in the project life cycle of Figure 2.6, the more accurate the information that can be used by the project planner. However, awaiting project execution and applying a purely reactive approach would result in unrealistic planning and ultimately in chaos, as described above. For risk management to make a real impact, it has to be present from the project initiation phase on. The project plan should be realistic. Proactivity is an absolute necessity.

In the iterative process of the project planning phase, we gathered all required information to be able to construct a proactive baseline schedule. In accordance with the CC/BM methodology, this schedule will be generated by using the expected activity durations d_j, but now these 50% confidence durations are the product of extensive risk management rather than subjective estimates. The proactive scheduling procedures introduced

in Chapters 4, 5 and 6 can all be used, together with many project scheduling methods proposed in the project scheduling literature or by commercial software packages. Evaluation of a candidate baseline schedule is possible by Monte-Carlo simulation. Instances of d_j are drawn from the triangular distribution defined by lower limit a_j, mode c_j and upper limit b_j. A simple reactive procedure decides how to repair the baseline schedule on schedule breakage during simulation. Simulative evaluation of several candidate baseline plans will result in the detection of the strengths and weaknesses of the candidates and might help the project manager to select a suitable project baseline schedule.

The proposed baseline plan should still be approved by project managers. A feasibility check is crucial. Additional organizational constraints may still be required at this point. It should also be verified whether the project objectives are obtained. Chapters 3 and 8 showed the importance of the project due date and the cost of exceeding the project due date (w_n) on the trade-off between stability and makespan. Although the due date is mostly established contractually, the above-described simulation approach might uncover that the predefined due date is unrealistic. Mostly the probability that the project finishes in time will be extremely small. To obtain credibility, a project due date prolongation should be considered.

The output of this phase is a well-thought-out baseline schedule that will hopefully decrease the schedule nervousness.

9.2.4 Project control phase

Risk management does not end with the actual start of the project. The project in progress should be monitored closely and disruptions should be detected, analyzed and stored in the risk management database (for future projects).

The actual progress of the project should be compared with the planned progress and if this reveals that the project is on its way to anarchy, corrective actions need to be taken. These actions might have already been recorded as risk treatments in the project planning phase or might be newly proposed disruption responses. An update of the baseline schedule into a workable projected schedule for the remainder of the project (see Chapter

7) might also be required.

What we often encounter in real-life projects is not a lack of inventiveness of project managers to find corrective actions, but rather an unorganized way of dealing with disruptions. We strongly advise project managers to plan their project until project completion. Reactiveness is an integral part of project scheduling.

9.2.5 Project termination phase

When objectives are not reached, budgets are exceeded or the due date is exceeded, upper management will certainly reprimand the responsible employees. But such an evaluation of tactical nature does not suffice.

Project managers are mostly relieved that a project is finished and forget to extract the valuable operational lessons that they can learn for future projects.

The strategic decisions made should be re-evaluated compared to their alternatives. The uniqueness property of a project makes this very hard. Estimated distribution functions of activity durations can for example not be proven to be wrong or right by one realized occurrence of the activity. The losses by including safety are also often more straightforward than the gains.

The risk management database should also be completed after the project, such that additional accurate historical data becomes available for future projects. Reusability of information is an absolute requirement to make risk management feasible.

9.3 Applying the framework to a real-life project

In this section, we document the application of our framework to a real-life project in the Belgian construction industry. The project at hand consists of the construction of an office building in Brussels. The building has multiple floors that are very similar, resulting in a repetitive project structure. We will illustrate our framework on one such floor, which is shown in Figure 9.4. This floor is constructed in two phases and four slots (A, B, C and D), which are bounded by four girders (O, R, T and U), can

be distinguished. We refer to Schatteman et al. (2006) for an illustration of the application of our risk management framework to a different large-scale project with well over 200 activities.

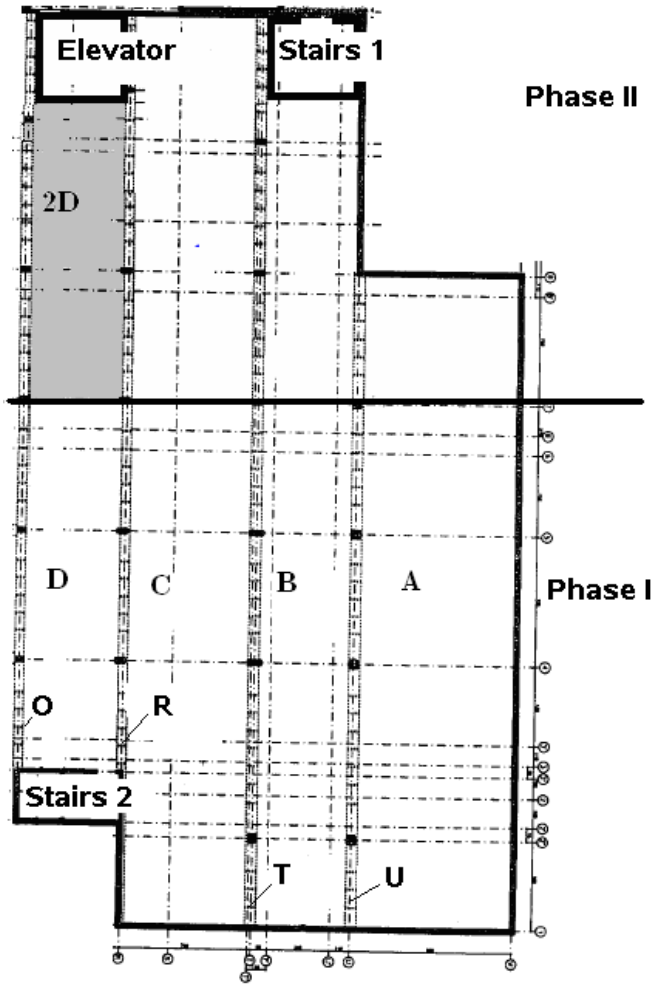


Figure 9.4: Layout of one floor in office building

9.3.1 The project network

In a first meeting with the project manager, the required data are collected and a project network is generated to avoid the problems discussed in Section 9.2.2.2. The resulting partial project network is shown in Figure 9.5.

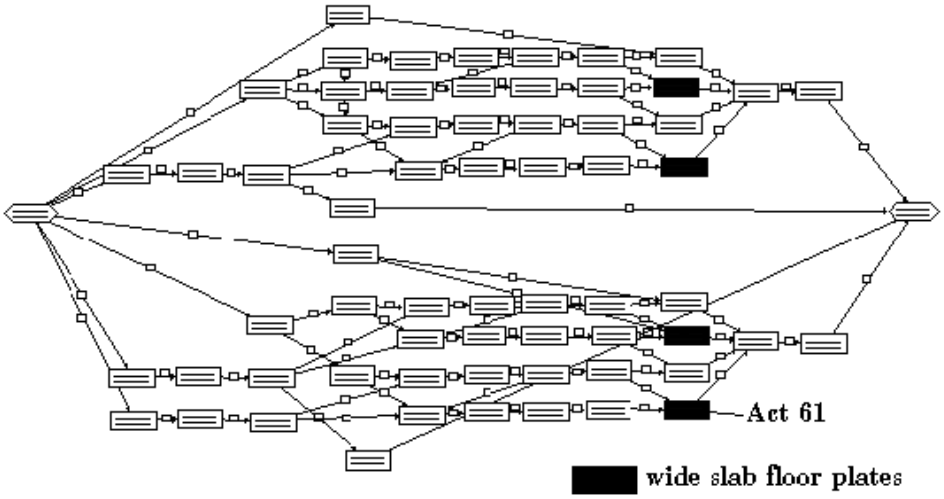


Figure 9.5: The project network

While common practice might be to work from left to right on such workplaces, activities from different slots in the layout are not precedence related in the network. The constructed network comprises flexibility for robust project scheduling procedures.

9.3.2 Risk detection & risk analysis

We illustrate our risk management approach on activity 61 in the project network. This activity is part of the activity group that is labeled as *wide slab floor plates*, which are pre-cast concrete slabs that are usually placed on girders. All four activities in this activity group have been colored black in the network of Figure 9.5. Activity 61 represents the placement of pre-cast floor plates in the slot of the building that has been indicated with label $2D$ in the layout of the office building (Figure 9.4).

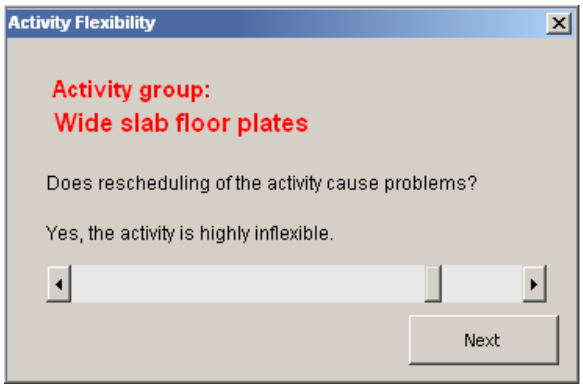
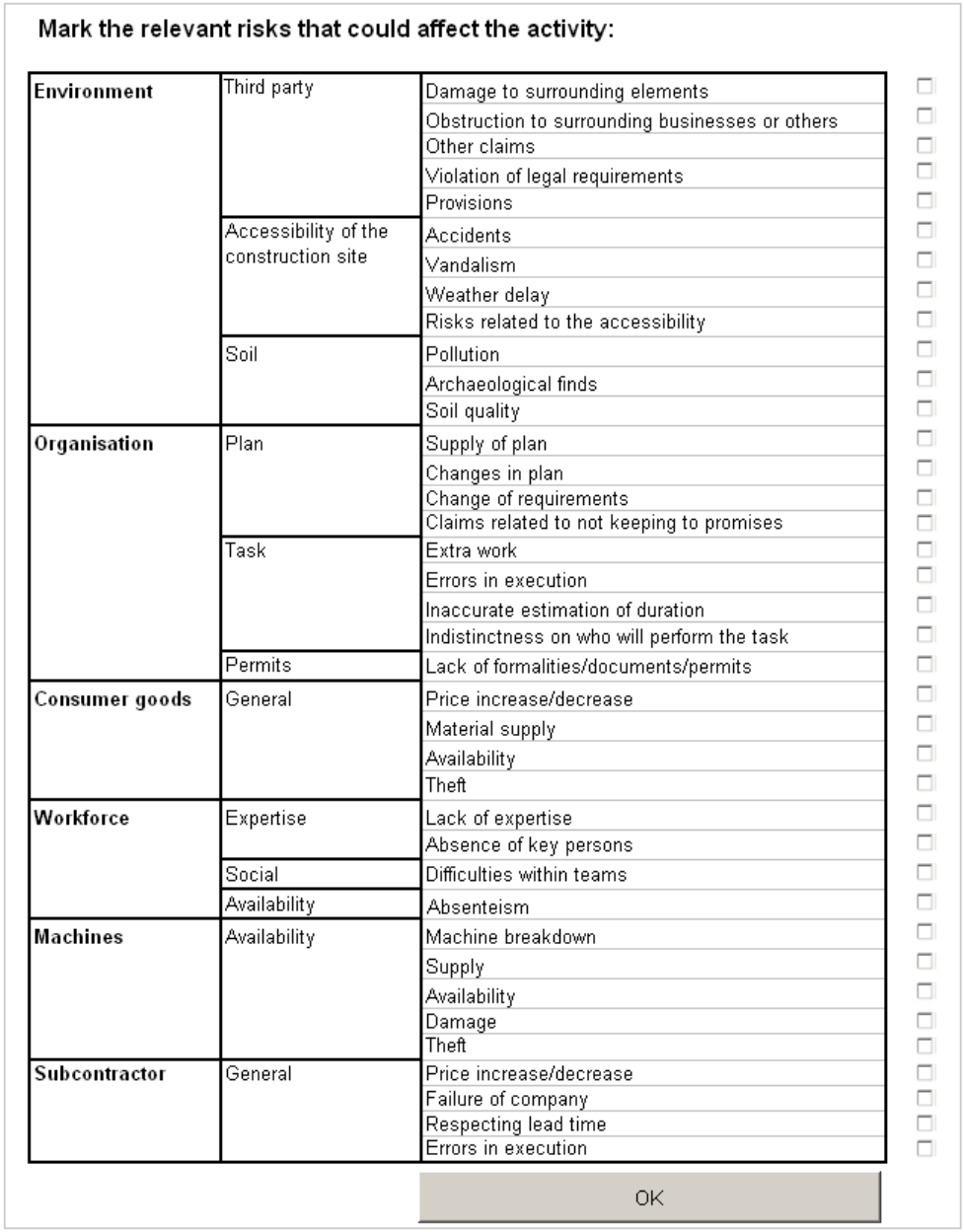


Figure 9.6: Interface to specify the flexibility of an activity

After detecting the activities and activity groups, the project manager is asked for the flexibility of (re)scheduling the activities in a given activity group by prompting a slider bar interface as shown in Figure 9.6. The slider bar indicates that the wide slab floor plates activity group is considered as rather inflexible. Wide slab floor plates are massive plates that need to be delivered by specialized transportation and need to be placed directly after arrival because storage is almost impossible. This requires the immediate availability of transport facilities and a hoisting crane. Renegotiating agreements for these resources with subcontractors in case of a delay are difficult.

Detecting the risks on the wide slab floor plate activity group is the next task for the project manager. An extensive checklist of possible risks is extracted from the risk management database and illustrated in Figure 9.7. Risks are divided in six main categories, i.e. environment, organization, materials, people, machines and subcontractors.

For the examined activity group in the current project, the project manager considered four risks to be important, i.e. bad weather conditions (environment), late delivery of acceptable plans (organization), absenteeism (workforce) and machine breakdowns (machines).



Mark the relevant risks that could affect the activity:

Environment	Third party	Damage to surrounding elements	☐
		Obstruction to surrounding businesses or others	☐
		Other claims	☐
		Violation of legal requirements	☐
		Provisions	☐
	Accessibility of the construction site	Accidents	☐
		Vandalism	☐
		Weather delay	☐
		Risks related to the accessibility	☐
	Soil	Pollution	☐
		Archaeological finds	☐
		Soil quality	☐
Organisation	Plan	Supply of plan	☐
		Changes in plan	☐
		Change of requirements	☐
		Claims related to not keeping to promises	☐
	Task	Extra work	☐
		Errors in execution	☐
		Inaccurate estimation of duration	☐
		Indistinctness on who will perform the task	☐
	Permits	Lack of formalities/documents/permits	☐
Consumer goods	General	Price increase/decrease	☐
		Material supply	☐
		Availability	☐
		Theft	☐
Workforce	Expertise	Lack of expertise	☐
		Absence of key persons	☐
	Social	Difficulties within teams	☐
	Availability	Absenteism	☐
Machines	Availability	Machine breakdown	☐
		Supply	☐
		Availability	☐
		Damage	☐
		Theft	☐
Subcontractor	General	Price increase/decrease	☐
		Failure of company	☐
		Respecting lead time	☐
		Errors in execution	☐

OK

Figure 9.7: Risk checklist for wide slab floor plates

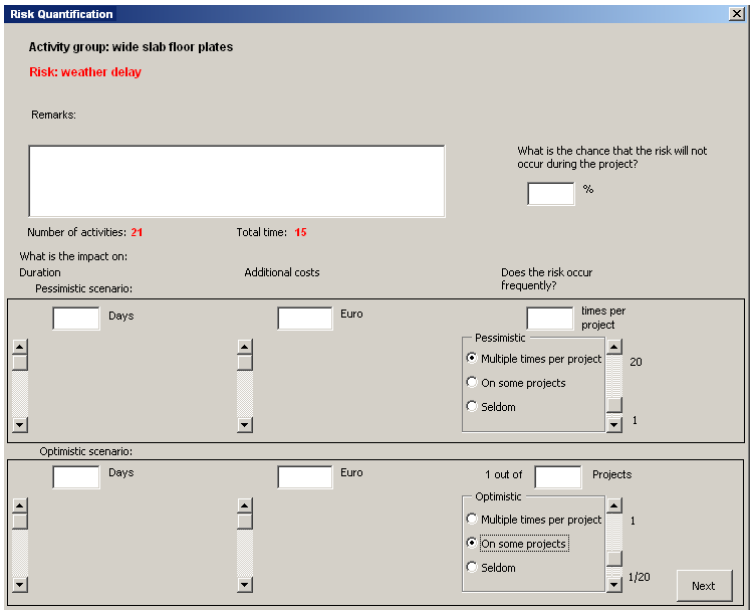


Figure 9.8: A graphical user interface for risk quantification

For each of these four risks, a scenario-based approach for risk analysis as proposed in Section 9.2.2.4 has to be applied. Estimates of the probability and the impact of both best and worst case scenario and the overall frequency of risk occurrence are asked during an interview and entered in a GUI[4] such as monitored in Figure 9.8. These estimates are transformed into a distribution function for the expected impact. Figure 9.9 shows the distribution function $f(\mathbf{x})$ of the expected delay on the wide slab floor plates activity group due to bad weather condition. A similar approach will supply distribution functions of the impact of the other risks on the wide slab floor plates activity group.

Subsequently, these impacts are mapped onto the activities within the activity group given the characteristics of the risks. By simulating a large number of project executions, a range of estimates for the realized activity duration d_{61} is obtained. The dashed line in Figure 9.10 shows the simulated distribution function of these estimates. The full line represents the fitted

[4]Remark that this graphical user interface provides functionalities such as cost analysis tools that are not discussed in this dissertation.

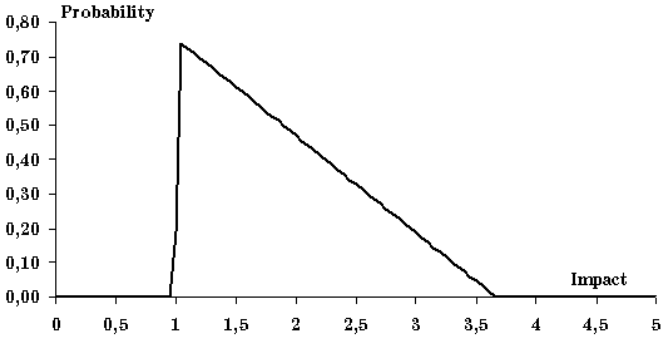


Figure 9.9: The impact of bad weather as a triangular distribution

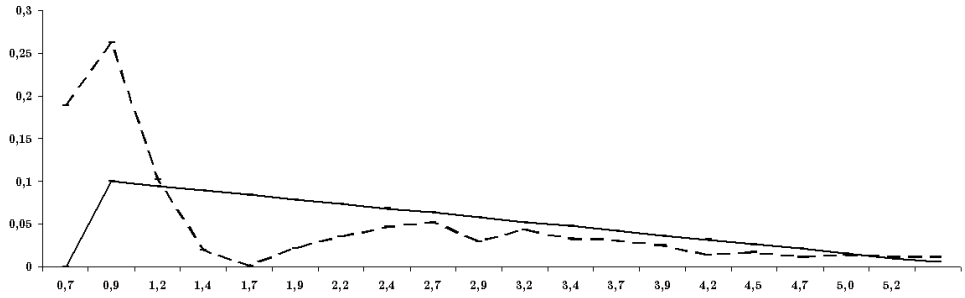


Figure 9.10: Triangular distribution function for $\mathbf{d}_{61}$

triangular distribution $f(\mathbf{d}_{61})$. From this triangular distribution, three estimates of $\mathbf{d}_{61}$ are distilled, respectively an optimistic estimate $a_{61} = 1$, a pessimistic $b_{61} = 6$ and a most likely estimate $c_{61} = 1$.

9.3.3 Project schedule

Multiple candidate schedules can now be generated for this project by applying either procedures embedded in commercial software packages, RCPSP procedures or any of the procedures introduced in previous chapters of this dissertation. We schedule activities by employing their expected durations $E(\mathbf{d}_j)$ that can be calculated as:

$$E(\mathbf{d}_j) = \frac{a_j + b_j + c_j}{3}. \tag{9.12}$$

Based on the risks detected and analyzed previously, these schedules can be evaluated through simulation by drawing a number of activity duration scenarios from the triangular distributions defined by the parameters a_j, b_j and c_j. We decide on the resource allocation by running the procedure of Artigues et al. (2003) and react during simulation by applying a robust parallel schedule generation scheme to an activity list that orders the activities by increasing starting times in the baseline schedule. The values of the project due date δ_n and the weight of the last activity w_n are crucial for schedule evaluation. Simulation results and schedule acceptance highly depend on these parameter settings.

When the project is scheduled as a deterministic project with average activity durations by the standard RCPSP scheduling mechanism embedded in MS Project, we obtain the schedule of Figure 9.12 with a makespan of 36 working days. Because all activities are scheduled with 50% confidence durations, this makespan can by no means serve as a realistic due date for the project, independently of which scheduling procedure is used.

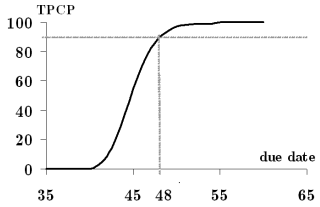


Figure 9.11: TPCP versus project due date

Figure 9.11 illustrates the simulated time project completion probability (TPCP) when working with the baseline schedule of Figure 9.12 as a function of the due date. We see that $P(\mathbf{s}_n^T \leq 36) = 0$. Based on the project managers statement that he wishes to obtain a 90% service level, it can be seen that the project due date should be set to at least 48 days later than the project start date. Hence, a 12-day project buffer is included to anticipate the variability of the complete project. The project is scheduled from August 1st 2006 until October 6th. Weekends are excluded from consideration.

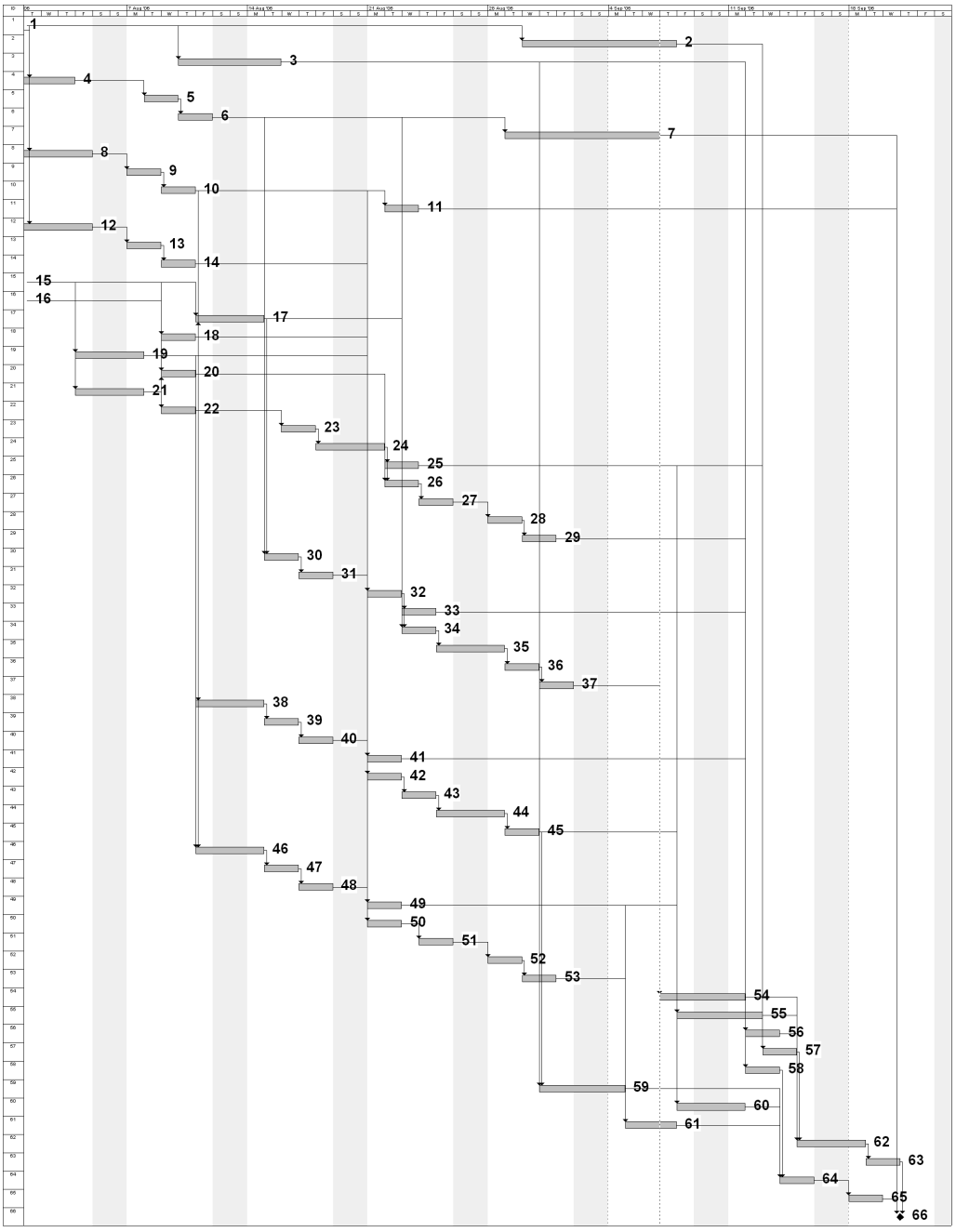


Figure 9.12: Project schedule obtained by MS Project

Simulation reveals that given the risks analyzed previously, the average due date excess $E|s_n^T - \delta_n|$ above the 48 days due date is 0.2 days. This results in an average stability cost for the last dummy activity that equals $w_n \times 0.2$. Simulated intermediate stability losses are 804 for this schedule, independently of the values of w_n and δ_n.

The average value for w_i selected by the project manager on the slider bars of Figure 9.6 for the activity groups in this project was approximately 3. Because the project completion was deemed important, we first set $w_n = 100$. This results in a stability cost of 824 for the quality robust schedule.

For a due date of 48 days, we are able to substantially reduce this stability cost by applying the predictive-reactive procedures introduced in this dissertation. We apply the STC heuristic of Section 5.1.3 on an initial schedule that has been generated by the ant optimization metaheuristic of Herbots et al. (2004) and obtain the schedule of Figure 9.13. Evaluating this schedule by simulation gives a stability cost of only 182. It is clear that this schedule is far more robust than the one depicted in Figure 9.12. However, the TPCP also decreased to 82% with an average due date excess of 0.38. If this service level is considered insufficient, robust project scheduling should not be rejected. Robust scheduling procedures are designed to attribute as much attention to project completion as they have been told to. A service level that is deemed inadequate, often indicates that the weight w_n has been underestimated. Setting $w_n = 500$ results in the schedule of Figure 9.14. The stability cost is 347, which is obviously still much better than the stability cost of the schedule of Figure 9.12, and the TPCP again equals 90%.

This schedule reincarnates the theory of the trade-off between makespan and stability in a real-life environment. A substantial improvement in solution robustness has been achieved without a concession on quality robustness. Working with this baseline schedule greatly improves the stability during project execution.

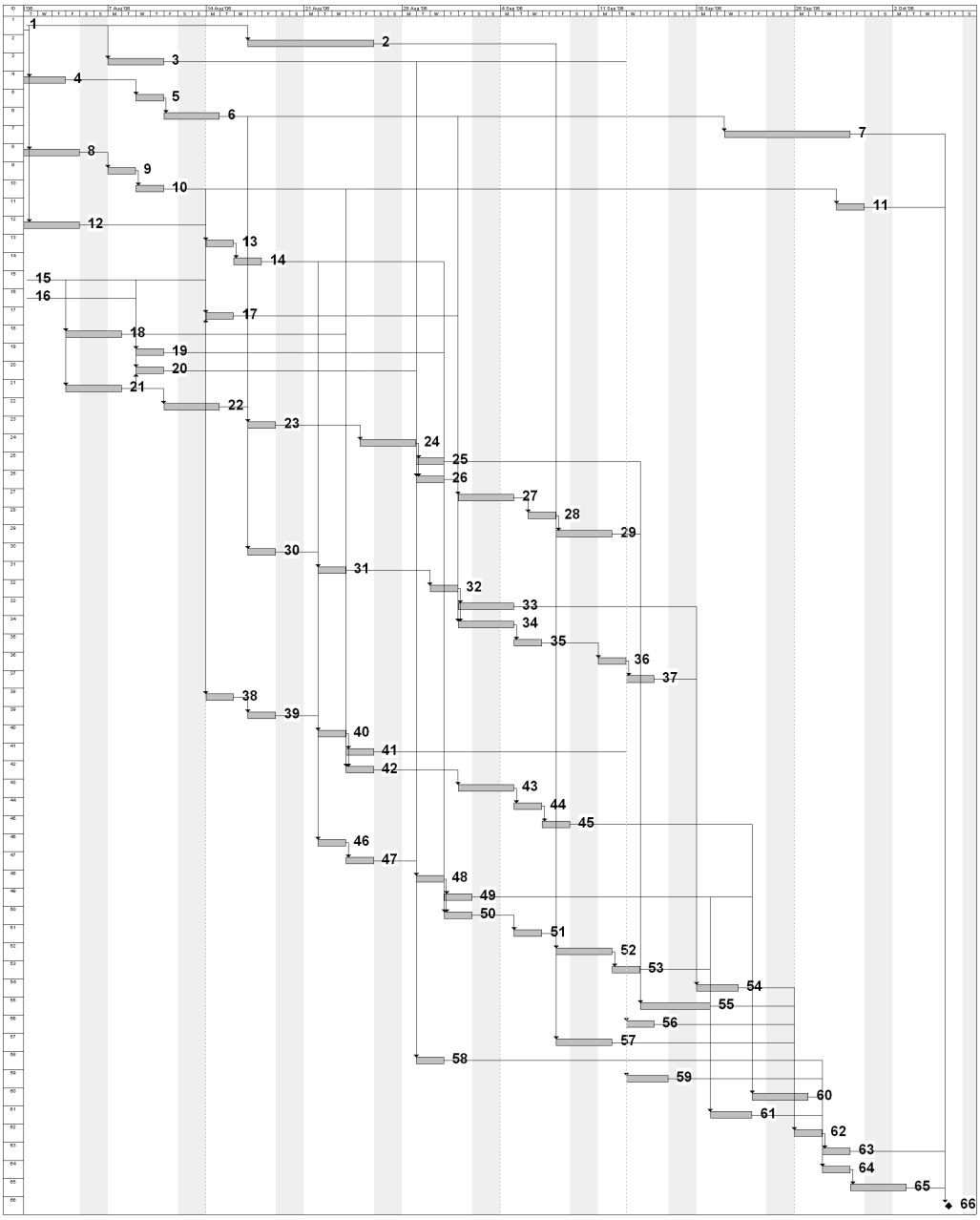


Figure 9.13: Baseline schedule obtained by STC when $w_n = 100$

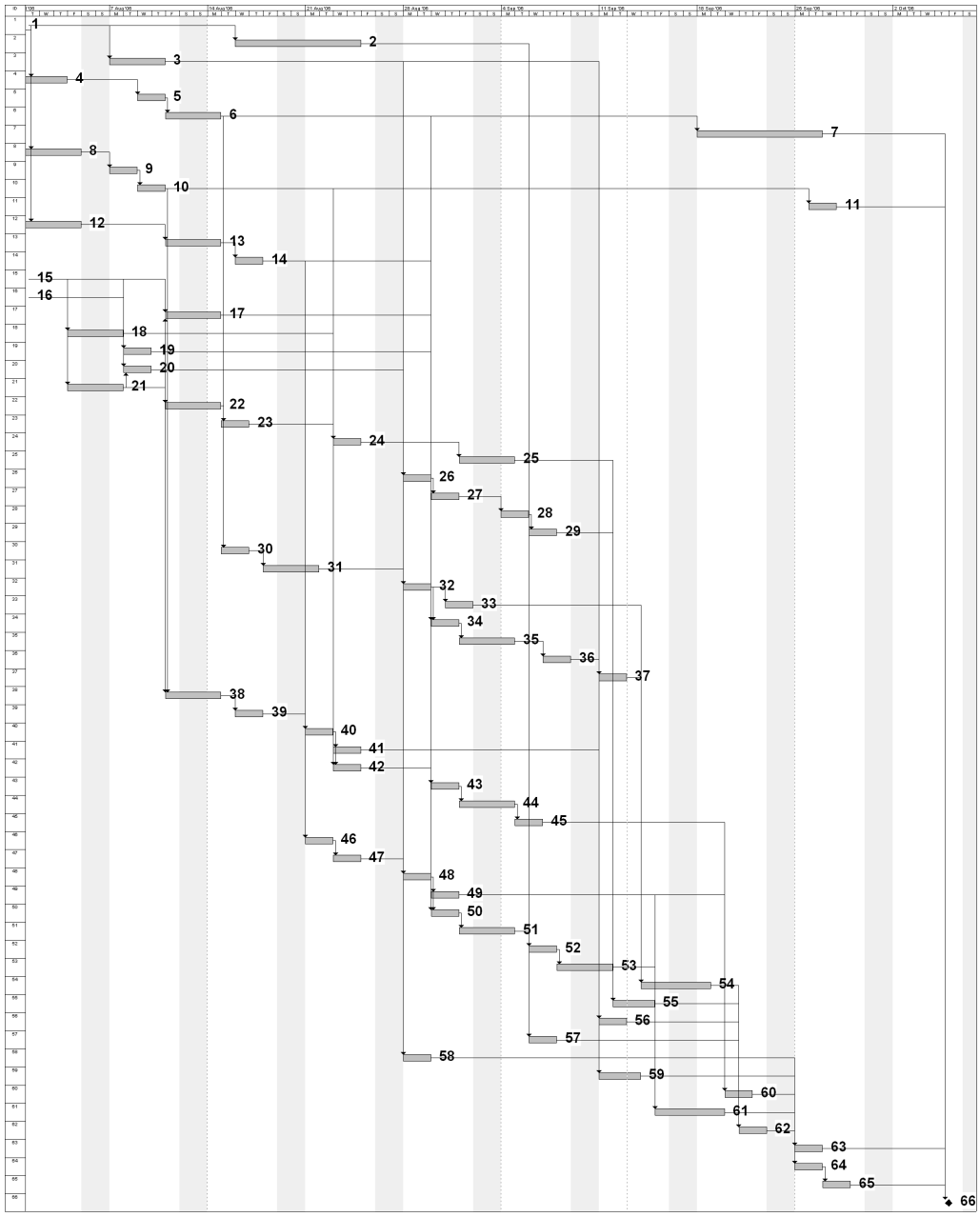


Figure 9.14: Baseline schedule obtained by STC when $w_n = 500$

We observe that our example activity number 61 is scheduled to start on Friday 15 September 2006 and has an expected duration of three days. Following the resource flow network, activity 61 can only start whenever activity 59 ends, because the same resources are required. Activity 59 is scheduled to be executed from Monday 11 September until Wednesday 13 September. Thursday September 14th is used as buffer for possible disruptions. Because activity 59 is also a member of the wide slab floor plates activity group, it is subject to the same risks as activity 61. Rain for a complete day between Monday 11 and Wednesday 13 will not affect the project execution. When working with the schedule of Figure 9.12, such a disruptive event would directly require to repair the project schedule and to renegotiate the agreements with some subcontractors.

9.4 Communication and acceptance

The need for a robust project baseline schedule as a basis for internal planning is a valuable tool for project management and top management within the organization. However, communicating such a robust project plan to external parties can cause some problems. In this section we will address some communication and acceptance issues that we have crossed during our experiences in the construction industry.

9.4.1 Internal acceptance

This dissertation has proposed many reasons why risk management and solution robust project planning could help the manager to improve the quality of the project. Still, many project managers are reluctant to plan their projects in such a way, because they claim that including safety is simply impossible for their project. We have to contest this claim by stating that every realistic project schedule has inherent safety, but that this safety is mostly handled unconsciously. Activity durations that are used to schedule the project are in practice commonly time windows that correspond to 80%-90% confidence intervals of the stochastic activity durations. The excess time above the 50% confidence level can thus be seen as an activity buffer. The buffer sizing happens rather subjectively and unconsciously. Our

framework helps at making uncertainty anticipation a conscious process. Such an approach reduces the nervousness during project execution and leads to a much more stable and realistic project schedule.

9.4.2 Internal communication

The communication between the project management and the project workforce might be an issue in many real-life projects. The CC/BM methodology unveils the prerequisite to avoid project delays caused by Parkinson's Law (Parkinson 1957). This law is also often referred to as the student's syndrome and denotes that, much as students do, the workforce responsible for executing a project activity tends to use the complete time window allowed. It can thus be seen as an absolute requirement that the buffers included in a robust project schedule to anticipate possible risks, are not communicated to the workforce. Otherwise, the work would expand to fill the time window plus buffers and a disruption near the end of the activity would cause a project delay because no more safety is available

9.4.3 External acceptance and communication

In the Belgian construction industry, most projects are granted by auctions. As long as the auctioneer is not convinced of the advantages of robust project planning strategies, communicating these strategies to them could damage the organization's chances to secure the project. This leads to a situation were an unrealistic project plan with tight budget and due date has to be communicated in order to acquire projects. At the end, the organization that is willing to take the largest risks is going to be the threshold. Obviously, such a situation is completely opposite to what we try to obtain.

In the light of the risk management project, improving awareness of the need for risk management is one of our main future challenges. Our risk management work group will be enlarged by including project auctioneers and government representatives. The ultimate aim is to obtain a standard for risk management such that the quality of project management would improve and that an organization that is convinced of the benefits of risk management could fully exploit them.

9.5 Conclusions and future research

In this chapter we attempted to narrow the gap between theory and practice in project scheduling. We introduced a framework that can help project managers to plan their project in a more robust way. Risk management is emphasized as a critical component of project management. We propose an approach to detect and analyze risks with a focus on ease of use, acceptability and clarity.

In a second part of this chapter, a real-life project served as an illustration for our methodological framework. Risk management and robust scheduling have recently been applied to several projects in the Belgian construction industry and many more are on-going. Very promising results have been achieved. We do still have a long way to walk before robust project scheduling will become common practice. Awareness and acceptance by all project stakeholders are the largest obstacles to overcome.

Some limitations of our procedures are being subject to future research to make them applicable to a wider range of projects. Future research directions include the extension to generalized precedence relationships (Elmaghraby & Kamburowski 1992); inclusion of specific techniques to model weather, machine breakdowns (Lambrechts et al. 2006a), spatial resources (de Boer 1998), weekend buffers[5] and activity delays (rather than activity duration prolongations). Narrowing the gap between practice and theory is a continuous process of mutual learning between both worlds. The work presented in this chapter has initiated this process.

[5]It is common practice in building industry to treat weekends as buffers for uncertainty.

Chapter 10

Conclusions

This final chapter will summarize the main conclusions drawn from the research efforts made in this thesis. We aim to give recommendations to both project managers and researchers interested in robust project scheduling.

Uncertainty lies at the heart of real-life project scheduling. We attempt to establish recognition of the impact of uncertainty on project planning. As for most problems that a manager will ever encounter, project scheduling under uncertainty can be tackled by either a proactive, a reactive or a proactive-reactive approach. We elaborated on the trade-off between the different options.

Chapter 3 revealed the importance of cleverly dealing with uncertainty for a project. The study of the trade-off between makespan and stability has demonstrated that for most projects a substantial gain in stability can be obtained while keeping the project service level to an acceptable level. A first paradoxical conclusion revealed that solution robust scheduling becomes particularly interesting for projects for which on time project completion is assumed to be essential.

In Chapters 4, 5 and 6 numerous proactive procedures have been proposed to improve the robustness of a baseline schedule. The followed approach is two-staged such that it starts by solving the basic resource-constrained project scheduling problem and adds safety to this initial schedule in a subsequent stage. This allows that the current project schedule within the organization can be transformed in a more robust version with-

out disregarding the efforts made to generate the original schedule. Improvements in stability can be obtained by either smart resource allocation (Chapter 4), buffer insertion (Chapter 5) or initial schedule selection (Chapter 6).

Several solution robust resource allocation procedures have been proposed in the project scheduling literature. Most of them obtain either unsatisfying results because of the use of an inferior surrogate objective function or are computationally demanding for large networks because they rely on solving large mixed integer programming formulations. We have introduced the MABO heuristic in Chapter 4, which requires little computational time to obtain satisfying results. Although we observed in Chapters 5 and 8 that the improvements of MABO are largely neutralized after buffer insertion; its computational efficiency leaves few practical objections against the use of such solution robust resource allocation procedures

The improvements on stability obtained in Chapter 4 are relatively small compared to the possible improvements by inserting time buffers in the initial schedule. An examination of several buffer insertion heuristics in Chapter 5 reveals that the activity dependent weights and the amount of variability present in the activity durations of the predecessors both influence the recommendable time window provided to complete an activity. STC was shown to be a relatively fast heuristic that performs well on small and large projects. Improvement algorithms and exact algorithms require simulation to evaluate several candidate solutions and become computationally infeasible for large scale projects.

The first stage of the two-stage approach, i.e. initial schedule selection, is commonly overlooked in robust project scheduling. We show in Chapter 6 that initial schedule selection has a substantial impact on solution robustness. However, selecting a solution robust initial schedule from several candidate schedules is a difficult task. The solution robustness of a candidate initial schedule can only be correctly evaluated after buffer insertion. In the second part of Chapter 6, a metaheuristic is proposed that abandons the two-stage approach and builds a stable buffered project schedule from scratch in one integrated stage. Despite the promising results obtained by this approach, it might suffer from acceptability issues in a real-life environ-

ment and will typically require much computational time before satisfying results are obtained.

The reactive procedures of Chapter 7 define how to react when disruptions occur such that the robustness introduced by the proactive procedure seems to be inadequate. The point of developing solution robust reactive procedures is two-fold. First, during actual project execution a project manager has to react to occurring disruptions. Repairing the original baseline schedule as good as possible often is a major aim during execution because this baseline schedule has been used as the basis for internal and external planning and communication. Second, most procedures proposed in Chapters 4, 5 and 6 heavily rely on simulation to compare the performance of multiple candidate baseline schedules. Reactive procedures are required to react to the simulated disruptions. The main difference between actual project execution and simulated project execution is that the former implies one unique execution of the unique baseline schedule, while the latter requires many simulated repetitions of several candidate schedules. Evaluation by simulation thus entails the need for fast reactive procedures.

Chapters 3-7 reveal that robust scheduling certainly pays-off. In Chapter 8 a large-scale computational experiment is set-up to compare the magnitude of the improvements obtained by the approaches developed in the different chapters. Buffer insertion (Chapter 5) obviously has the largest positive impact on stability. Besides, buffer insertion procedures start from a given initial schedule and a fixed resource flow network and largely manage to neutralize the stability improvements obtained by initial schedule selection or clever resource allocation.

However, the negative impact of buffer insertion on makespan performance might still make it hard to convince project managers to apply such an approach. Certainly for projects in a highly variable environment and with a rather tight due date, it becomes challenging to improve solution robustness by idle time insertion without compromising on quality robustness. For such projects, solution robust resource allocation techniques or clever schedule repair procedures might substantially improve the project's stability. This induces a second paradoxical conclusion, namely the more uncertainty inherent to a project, the less powerful procedures can be applied

to combat stability costs.

Chapter 9 applies the procedures of the previous chapters to a real-life project. We introduce a framework for risk management that is able to detect, analyze and quantify the uncertainty in a project. Improved awareness of the risks and opportunities leads to the possibility to deal with them. The field of operations management can greatly assist project managers in this matter. Unique characteristics of every project make universal rules impossible, but a better awareness of the trade-offs present in every scheduling decision can undoubtedly improve the quality of a project.

Although we attempted to narrow the gap between operations research models and their real-life application, this gap will always persist. Despite the fact that most operations research tools will most probably experience resistance to be implemented in business environments, operations researchers can already make a valid contribution by allowing practitioners to reflect on their traditional decision processes. Even if practitioners are reluctant to use the newly developed operations research tools, they will undoubtedly influence their ways of thinking and acting.

List of Figures

List of Tables

Bibliography

Al-Fawzan, M. A. & Haouari, M. (2005). A bi-objective model for robust resource-constrained project scheduling. *International Journal of Production Economics*, 96(2), pp 175–187.

Aloulou, M. & Portmann, M. (2003). An efficient proactive reactive scheduling approach to hedge against shop floor disturbances. *1st Multidisciplinary Conference on Scheduling: Theory and Applications (MISTA 2003)*. pp 337–362.

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

Artigues, C., Michelon, P. & Reusser, S. (2003). Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*, 149(2), pp 249–267.

Artigues, C. & Roubellat, F. (2000). A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes. *European Journal of Operational Research*, 127, pp 294–316.

Aytug, H., Lawley, M., McKay, K., Mohan, S. & Uzsoy, R. (2005). Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operational Research*, 161(1), pp 86–110.

Ballestín, F. (2006). When it is worthwhile to work with the stochastic RCPSP? *Journal of Scheduling,* to appear.

Ballestín, F. & Leus, R. (2006). Meta-heuristics for stable scheduling on a single machine. *Computers & Operations Research,* to appear.

Ballestín, F. & Trautmann, N. (2006). A metaheuristic approach for the resource-constrained weighted earliness-tardiness project scheduling problem. Working paper.

Billaut, J.-C., Moukrim, A. & Sanlaville, E. (eds) (2005). *Flexibilité et robustesse en ordonnancement*. Traité IC2, Série Informatique et systèmes d'information. Hermès-Lavoisier.

Blazewicz, J., Cellary, W., Slowinski, R. & Weglarz, J. (1986). Scheduling under resource constraints - deterministic models. *Annals of Operations Research*, 7, pp 1–359.

Blazewicz, J., Lenstra, J. & Kan, A. R. (1983). Scheduling subject to resource constraints - classification and complexity. *Discrete Applied Mathemetics*, 5, pp 11–24.

Brucker, P., Drexl, A., Möhring, R., Neumann, K. & Pesch, E. (1999). Resource-constrained project scheduling: notation, classification, models and methods. *European Journal of Operational Research*, 112, pp 3–41.

Brucker, P. & Knust, S. (2006). *Complex Scheduling*. Springer.

Cesta, A., Oddi, A. & Smith, S. (1998). Profile-based algorithms to solve multi-capacitated metric scheduling problems. *Proceedings 5th International Conference on Artificial Intelligence Planning Systems*.

Chapman, C. & Ward, S. (2003). *Project Risk Management. Process, techniques and insights*. John Wiley & Sons, Ltd.

Cooper, D. (1976). Heuristics for scheduling resource constrained projects: an experimental investigation. *Management Science*, 22, pp 1186–1194.

de Boer, R. (1998). *Resource-Constrained Multi-Project Management - A Hierarchical Decision Support System*. PhD thesis. University of Twente, The Netherlands.

Debels, D. & Vanhoucke, M. (2006). Future research avenues for resource constrained project scheduling: Search space restriction or neighbourhood search extension. *Research report*. Ghent University, Belgium.

Deblaere, F., Demeulemeester, E., Herroelen, W. & Van de Vonder, S. (2006). Proactive resource allocation heuristics for robust project scheduling. *Research Report 0608*. Department of decision sciences and information management, Katholieke Universiteit Leuven.

Demeulemeester, E. & Herroelen, W. (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, 38, pp 1803–1818.

Demeulemeester, E. & Herroelen, W. (1997). New benchmark results for the resource-constrained project scheduling problem. *Management Science*, 43, pp 1485–1492.

Demeulemeester, E. & Herroelen, W. (2002). *Project scheduling - A research handbook*. Vol. 49 of *International Series in Operations Research & Management Science*. Kluwer Academic Publishers, Boston.

Demeulemeester, E., Vanhoucke, M. & Herroelen, W. (2003). RanGen: A random network generator for activity-on-the-node networks. *Journal of Scheduling*, 6, pp 17–38.

Dodin, B. (2006). *A practical and accurate alternative to PERT*. in Weglarz J, Jozefowska J, ed.: Perspectives in Modern Scheduling, (1).

Drexl, A. (1991). Scheduling of project networks by job assignment. *Management Science*, 37, pp 1590–1602.

Drezet, L. (2005). *Résolution d'un problème de gestion de projets sous contraintes de resources humaines: de l'approche prédictive l'approche réactive*. PhD thesis. Universit François Rabelais Tours.

Elmaghraby, S. (2005). On the fallacy of averages in project risk management. *European Journal of Operational Research*, 165(2), pp 307–313.

Elmaghraby, S. & Kamburowski, J. (1992). The analysis of activity networks under generalized precedence relations GPRs. *Management science*, 38(9), pp 1245–1263.

Fernandez, A. A. & Armacost, R. L. (1995). The role of the non-anticipativity constraint in commercial software for stochastic project scheduling. *Computers and Industrial Engineering*, 31, pp 233–236.

Fulkerson, D. (1961). A network flow computation for project cost curves. *Management science*, 7(2), pp 167–178.

Glover, F. (1989). Tabu search, Part I. *INFORMS, Journal of Computing*, 1, pp 190–206.

Glover, F. (1990). Tabu search, Part II. *INFORMS, Journal of Computing*, 2, pp 4–32.

Goldratt, E. (1997). *Critical Chain*. The North River Press Publishing Corporation, Great Barrington.

Golenko-Ginzburg, D. & Gonik, A. (1998). A heuristic for network project scheduling with random activity durations depending on the resource allocation. *International Journal on Production Economics*, 55, pp 149–162.

Graham, R. L. (1966). Bounds on multiproccessing timing anomalies. *Bell System Technical Journal*, 45, pp 1563–1581.

Hagstrom, J. (1988). Computational complexity of PERT problems. *Networks*, 18, pp 139–147.

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics*, 45(7), pp 733–750.

Herbots, J., Herroelen, W. & Leus, R. (2004). Experimental investigation of the applicability of ant colony optimization algorithms for project scheduling. *Research Report 0459*. Department of applied economics, Katholieke Universiteit Leuven, Belgium.

Herroelen, W., De Reyck, B. & Demeulemeester, E. (1998). Resource-constrained scheduling: a survey of recent developments. *Computers and Operations Research*, 25, pp 279–302.

Herroelen, W., De Reyck, B. & Demeulemeester, E. (2000). On the paper "Resource-constrained project scheduling: notation, classification, models and methods" by Brucker et al.. *European Journal of Operational Research*, 128(3), pp 221–230.

Herroelen, W. & Leus, R. (2001). On the merits and pitfalls of critical chain scheduling. *Journal of Operations Management*, 128(3), pp 221–230.

Herroelen, W. & Leus, R. (2004). The construction of stable baseline schedules. *European Journal of Operational Research*, 156, pp 550–565.

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

Hoogeveen, H. (2005). Multicriteria scheduling. *European Journal of Operational Research*, 167(3), pp 592–623.

Igelmund, G. & Rademacher, F. (1983). Preselective strategies for the optimization of stochatic project networks under resoource constraints. *Networks*, 13, pp 1–28.

Kéri, A. & Kis, T. (2005). Primal-dual combined with constraint propagation for solving rcpspwet. *Proceedings of 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications, New York.*

Knight, F. (1921). *Risk, Uncertainty, and Profit.* Boston, MA: Hart, Schaffner & Marx; Houghton Mifflin Company.

Kolisch, R. (1996a). Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management*, 14, pp 179–192.

Kolisch, R. (1996b). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 90, pp 320–333.

Kolisch, R. & Hartmann, S. (1999). *Heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis*. Kluwer Academic Publishers. In Weglarz J.: Project scheduling: recent models, algorithms and applications.

Kolisch, R. & Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 174, pp 23–37.

Kolisch, R. & Padman, R. (1999). An integrated survey of deterministic project scheduling. *Omega*, 49, pp 249–272.

Kolisch, R. & Sprecher, A. (1997). PSPLIB - a project scheduling library. *European Journal of Operational Research*, 96, pp 205–216.

Kouvelis, P. & Yu, G. (1997). *Robust discrete optimization and its applications*. Kluwer Academic Publishers, Boston.

Lambrechts, O., Demeulemeester, E. & Herroelen, W. (2006a). Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. *Research report*. Department of decision sciences and information management, Katholieke Universiteit Leuven.

Lambrechts, O., Demeulemeester, E. & Herroelen, W. (2006b). A tabu search procedure for generating robust project baseline schedules under stochastic resource availabilities. *Research report*. Department of decision sciences and information management, Katholieke Universiteit Leuven.

Leach, L. (2000). *Critical chain project management*. Artec house professional development library.

Leon, V., Wu, S. D. & Storer, R. H. (1994). Robustness measures and robust scheduling for job shops. *IIE Transactions*, 16(5), pp 32–43.

Leus, R. (2003). *The generation of stable project plans*. PhD thesis. Department of applied economics, Katholieke Universiteit Leuven, Belgium.

Leus, R. & Herroelen, W. (2004). Stability and resource allocation in project planning. *IIE transactions*, 36(7), pp 1–16.

Lourenço, H. R., Martin, O. & Stützle, T. (2002). *Iterated Local Search*. Kluwer, Boston. pp 321–353. In F. Glover and G. Kochenberger (eds.): Handbook of metaheuristics.

Ludwig, A., Mhring, R. & Stork, F. (2001). A computational study on bounding the makespan distribution in stochastic project networks. *Annals of Operations Research*, 102(1/4), pp 49–64.

Maes, J., Vandoren, C., Sels, L. & Roodhooft, F. (2000). Onderzoek naar oorzaken van faillissementen van kleine en middelgrote bouwondernemingen. Unpublished manuscript, CTEO Leuven.

Mastor, A. (1970). An experimental and comparative evaluation of production line balancing techniques. *Management Science*, 16, pp 728–746.

Mehta, S. & Uzsoy, R. (1998). Predictive scheduling of a job shop subject to breakdowns. *IEEE Transactions on Robotics and Automation*, 14, pp 365–378.

Möhring, R., Radermacher, F. & Weiss, G. (1984). Stochastic scheduling problems I - Set strategies. *Zeitschrift für Operations Research*, 28, pp 193–260.

Möhring, R., Radermacher, F. & Weiss, G. (1985). Stochastic scheduling problems II - General strategies. *Zeitschrift für Operations Research*, 29, pp 65–104.

Newbold, R. (1998). *Project management in the fast lane - Applying the Theory of Constraints*. APICS Series on Constraints Management. The St. Lucie Press.

Parkinson, C. (1957). *Parkinson's law*. The riverside press, Cambridge.

Pascoe, T. (1966). Allocations of resources c.p.m. *Revue Franaise de Recherche Opratione lle*, 38, pp 31–38.

Patterson, J. (1976). Project scheduling: the effects of problem structure on heuristic scheduling. *Naval Research Logistics*, 23, pp 95–123.

Policella, N. (2005). *Scheduling with Uncertainty: A Proactive Approach using Partial Order Schedules*. PhD thesis. Università degli studi di Roma "La Sapienza".

Policella, N., Oddi, A., Smith, S. & Cesta, A. (2004). Generating robust partial order schedules. *In Proceedings of CP2004, Toronto, Canada*.

Radermacher, F. (1985). Scheduling of project networks. *Annals of Operations Research*, 4, pp 227–252.

Roy, B. (2002). Robustesse de quoi et vis-à-vis de quoi mais aussi robustesse pourquoi en aide à la décision?. *in* J. Figueira, C. Henggeler-Anthunes & J. Climaco (eds), *Proceedings of the 56th Meeting of the European Working Group on Multiple Criteria Decision Making, Coimbra*.

Sanlaville, E. (2004). *Ordonnancement sous conditions changeantes*. PhD thesis. Université Blaise Pascal, Clermont-Ferrand, France.

Schatteman, D., Herroelen, W., Van de Vonder, S. & Boone, A. (2006). A methodology for integrated risk management and proactive scheduling of construction projects. *Research report*. Department of decision sciences and information management, Katholieke Universiteit Leuven.

Schwindt, C. (2000). *Minimizing earliness-tardiness costs of resource-constrained projects*. Springer, Berlin. In Inderfurth K, Schwödiauer G, Domschke W, Juhnke F, Kleinschmidt P, Wäscher G (eds) Operations Research Proceedings.

Schwindt, C. (2005). *Resource allocation in project management*. GOR Publications. Springer.

Sörensen, K. (2001). Tabu searching for robust solutions. *Proceedings of the 4th Metaheuristics International Conference*.

Stork, F. (2001). *Stochastic Resource-Constrained Project Scheduling*. PhD thesis. Technical University of Berlin, School of Mathematics and Natural Sciences.

Tavares, L., Ferreira, J. & Coelho, J. (1998). On the optimal management of project risk. *European Journal of Operational Research*, 107, pp 451–469.

T'kindt, V. & Billaut, J.-C. (2006). *Multicriteria scheduling - Theory, Models and Algorithms*. Springer.

Tukel, O., Rom, W. & Eksioglu, S. (2006). An invvestigation of buffer size techniques in critical chain scheduling. *European Journal of Operational Research*, 172, pp 401–416.

Valls, V., Ballestín, F. & Quintanilla, S. (2002). A hybrid genetic algorithm for the resource-constrained project scheduling problem with the peak crossover operator. *Eighth International Workshop on Project Management and Scheduling*. pp 368–371.

Valls, V., Ballestín, F. & Quintanilla, S. (2005). Justification and RCPSP: a technique that pays. *European Journal of Operational Research*, 165, pp 375–386.

Valls, V., Laguna, M., Lina, P., Pérez, A. & Quintanilla, S. (1999). *Project Scheduling with Stochastic Activity Interruptions*. Kluwer Academic Publishers, Bostonk. pp 333–354. In: Weglarz, J. (Ed.), Project Scheduling. Recent Models, Algorithms and Applications.

Van de Vonder, S., Ballestín, F., Demeulemeester, E. & Herroelen, W. (2006a). Heuristic procedures for reactive project scheduling. *Computers & Industrial Engineering,* to appear.

Van de Vonder, S., Demeulemeester, E. & Herroelen, W. (2005a). Heuristic procedures for generating stable project baseline schedules. *European Journal of Operational Research,* to appear.

Van de Vonder, S., Demeulemeester, E. & Herroelen, W. (2005b). An investigation of efficient and effective predictive-reactive project scheduling procedures. *Journal of Scheduling,* to appear.

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

Van de Vonder, S., Demeulemeester, E., Herroelen, W. & Leus, R. (2006b). The trade-off between stability and makespan in resource-constrained project scheduling. *International Journal of Production Research*, 44(2), pp 215–236.

Van de Vonder, S., Demeulemeester, E., Leus, R. & Herroelen, W. (2006c). *Proactive/reactive project scheduling - Trade-offs and procedures*. pp 25–51. in Jozefowska J, Weglarz J, ed.: Perspectives in Modern Project Scheduling, (2).

van Well-Stam, D., Lindenaar, F. & van Kinderen, S. (2003). *Risicomanagement voor projecten*. Het Spectrum.

Vanhoucke, M., Demeulemeester, E. & Herroelen, W. (2001). An exact procedure for the resource-constrained weighted earliness-tardiness project scheduling problem. *Annals of Operations Research*, 102, pp 179–196.

Vieira, G., Herrmann, J. & Lin, E. (2003). Rescheduling manufacturing systems: a framework of strategies, policies, and methods. *Journal of Scheduling*, 6(1), pp 35–58.

Wu, S., Storer, H. & Chang, P.-C. (1993). One-machine rescheduling heuristics with efficiency and stability as criteria. *Computers and Operations Research*, 20, pp 1–14.

Yang, K.-K. (1996). Effects of erroneous estimation of activity durations on scheduling and dispatching a single project. *Decision Sciences*, 27(2), pp 255–290.

Yu, G. & Qi, X. (2004). *Disruption Management - Framework, Models and Applications*. World Scientific, New Jersey.

Index

Doctoral Dissertations from the Faculty of Economic and Applied Economic Sciences

From August 1, 1971.

1. GEPTS Stefaan (1971)
   Stability and efficiency of resource allocation processes in discrete commodity spaces. Leuven, KUL, 1971. 86 pp.

2. PEETERS Theo (1971)
   Determinanten van de internationale handel in fabrikaten. Leuven, Acco, 1971. 290 pp.

3. VAN LOOY Wim (1971)
   Personeelsopleiding: een onderzoek naar investeringsaspekten van opleiding. Hasselt, Vereniging voor wetenschappelijk onderzoek in Limburg, 1971. VII, 238 pp.

4. THARAKAN Mathew (1972)
   Indian exports to the European community: problems and prospects. Leuven, Faculty of economics and applied economics, 1972. X,343 pp.

5. HERROELEN Willy (1972)
   Heuristische programmatie: methodologische benadering en praktische toepassing op complexe combinatorische problemen. Leuven, Aurelia scientifica, 1972. X, 367 pp.

6. VANDENBULCKE Jacques (1973)
   De studie en de evaluatie van data-organisatiemethodes en data-zoekmethodes. Leuven, s.n., 1973. 3 V.

7. PENNYCUICK Roy A. (1973)
   The economics of the ecological syndrome. Leuven, Acco, 1973. XII, 177 pp.

8. KAWATA T. Bualum (1973)
   Formation du capital d'origine belge, dette publique et stratégie du développement au Zaire. Leuven, KUL, 1973. V, 342 pp.

9. DONCKELS Rik (1974)
   Doelmatige oriëntering van de sectorale subsidiepolitiek in België: een theoretisch onderzoek met empirische toetsing. Leuven, K.U.Leuven, 1974. VII, 156 pp.

10. VERHELST Maurice (1974)
Contribution to the analysis of organizational information systems and their financial benefits. Leuven, K.U.Leuven, 1974. 2 V.

11. CLEMEUR Hugo (1974)
Enkele verzekeringstechnische vraagstukken in het licht van de nutstheorie. Leuven, Aurelia scientifica, 1974. 193 pp.

12. HEYVAERT Edward (1975)
De ontwikkeling van de moderne bank- en krediettechniek tijdens de zestiende en zeventiende eeuw in Europa en te Amsterdam in het bijzonder. Leuven, K.U.Leuven, 1975. 186 pp.

13. VERTONGHEN Robert (1975)
Investeringscriteria voor publieke investeringen: het uitwerken van een operationele theorie met een toepassing op de verkeersinfrastructuur. Leuven, Acco, 1975. 254 pp.

14. Niet toegekend.

15. VANOVERBEKE Lieven (1975)
Microeconomisch onderzoek van de sectoriële arbeidsmobiliteit. Leuven, Acco, 1975. 205 pp.

16. DAEMS Herman (1975)
The holding company: essays on financial intermediation, concentration and capital market imperfections in the Belgian economy. Leuven, K.U.Leuven, 1975. XII, 268 pp.

17. VAN ROMPUY Eric (1975)
Groot-Brittannië en de Europese monetaire integratie: een onderzoek naar de gevolgen van de Britse toetreding op de geplande Europese monetaire unie. Leuven, Acco, 1975. XIII, 222 pp.

18. MOESEN Wim (1975)
Het beheer van de staatsschuld en de termijnstructuur van de intrestvoeten met een toepassing voor België. Leuven, Vander, 1975. XVI, 250 pp.

19. LAMBRECHT Marc (1976)
Capacity constrained multi-facility dynamic lot-size problem. Leuven, KUL, 1976. 165 pp.

20. RAYMAECKERS Erik (1976)
De mens in de onderneming en de theorie van het producenten-gedrag: een bijdrage tot transdisciplinaire analyse. Leuven, Acco, 1976. XIII, 538 pp.

21. TEJANO Albert (1976)
Econometric and input-output models in development planning: the case of the Philippines. Leuven, KUL, 1976. XX, 297 pp.

22. MARTENS Bernard (1977)
Prijsbeleid en inflatie met een toepassing op België. Leuven, KUL, 1977. IV, 253 pp.

23. VERHEIRSTRAETEN Albert (1977)
Geld, krediet en intrest in de Belgische financiële sector. Leuven, Acco, 1977. XXII, 377 pp.

24. GHEYSSENS Lieven (1977)
International diversification through the government bond market: a risk-return analysis. Leuven, s.n., 1977. 188 pp.

25. LEFEBVRE Chris (1977)
Boekhoudkundige verwerking en financiële verslaggeving van huurkooptransacties en verkopen op afbetaling bij ondernemingen die aan consumenten verkopen. Leuven, KUL, 1977. 228 pp.

26. KESENNE Stefan (1978)
Tijdsallocatie en vrijetijdsbesteding: een econometrisch onderzoek. Leuven, s.n., 1978. 163 pp.

27. VAN HERCK Gustaaf (1978)
Aspecten van optimaal bedrijfsbeleid volgens het marktwaardecriterium: een risico-rendements-analyse. Leuven, KUL, 1978. IV, 163 pp.

28. VAN POECK Andre (1979)
World price trends and price and wage development in Belgium: an investigation into the relevance of the Scandinavian model of inflation for Belgium. Leuven, s.n., 1979. XIV, 260 pp.

29. VOS Herman (1978)
De industriële technologieverwerving in Brazilië: een analyse. Leuven, s.n., 1978. onregelmatig gepagineerd.

30. DOMBRECHT Michel (1979)
Financial markets, employment and prices in open economies. Leuven, KUL, 1979. 182 pp.

31. DE PRIL Nelson (1979)
Bijdrage tot de actuariële studie van het bonus-malussysteem. Brussel, OAB, 1979. 112 pp.

32. CARRIN Guy (1979)
Economic aspects of social security: a public economics approach. Leuven, KUL, 1979. onregelmatig gepagineerd

33. REGIDOR Baldomero (1979)
An empirical investigation of the distribution of stock-market prices and weak-form efficiency of the Brussels stock exchange. Leuven, KUL, 1979. 214 pp.

34. DE GROOT Roger (1979)
Ongelijkheden voor stop loss premies gebaseerd op E.T. systemen in het kader van de veralgemeende convexe analyse. Leuven, KUL, 1979. 155 pp.

35. CEYSSENS Martin (1979)
On the peak load problem in the presence of rationizing by waiting. Leuven, KUL, 1979. IX, 217 pp.

36. ABDUL RAZK Abdul (1979)
Mixed enterprise in Malaysia: the case study of joint venture between Malysian public corporations and foreign enterprises. Leuven, KUL, 1979. 324 pp.

37. DE BRUYNE Guido (1980)
Coordination of economic policy: a game-theoretic approach. Leuven, KUL, 1980. 106 pp.

38. KELLES Gerard (1980)
Demand, supply, price change and trading volume on financial markets of the matching-order type. = Vraag, aanbod, koersontwikkeling en omzet op financiële markten van het Europese type. Leuven, KUL, 1980. 222 pp.

39. VAN EECKHOUDT Marc (1980)
De invloed van de looptijd, de coupon en de verwachte inflatie op het opbrengstverloop van vastrentende finaciële activa. Leuven, KUL, 1980. 294 pp.

40. SERCU Piet (1981)
Mean-variance asset pricing with deviations from purchasing power parity. Leuven, s.n., 1981. XIV, 273 pp.

41. DEQUAE Marie-Gemma (1981)
Inflatie, belastingsysteem en waarde van de onderneming. Leuven, KUL, 1981. 436 pp.

42. BRENNAN John (1982)
An empirical investigation of Belgian price regulation by prior notification: 1975 - 1979 - 1982. Leuven, KUL, 1982. XIII, 386 pp.

43. COLLA Annie (1982)
Een econometrische analyse van ziekenhuiszorgen. Leuven, KUL, 1982. 319 pp.

44. Niet toegekend.

45. SCHOKKAERT Eric (1982)
Modelling consumer preference formation. Leuven, KUL, 1982. VIII, 287 pp.

46. DEGADT Jan (1982)
Specificatie van een econometrisch model voor vervuilingsproblemen met proeven van toepassing op de waterverontreiniging in België. Leuven, s.n., 1982. 2 V.

47. LANJONG Mohammad Nasir (1983)
A study of market efficiency and risk-return relationships in the Malaysian capital market. s.l., s.n., 1983. XVI, 287 pp.

48. PROOST Stef (1983)
De allocatie van lokale publieke goederen in een economie met een centrale overheid en lokale overheden. Leuven, s.n., 1983. onregelmatig gepagineerd.

49. VAN HULLE Cynthia (1983)
Shareholders' unanimity and optimal corporate decision making in imperfect capital markets. s.l., s.n., 1983. 147 pp. + appendix.

50. VAN WOUWE Martine (2/12/83)
Ordening van risico's met toepassing op de berekening van ultieme ruïnekansen. Leuven, s.n., 1983. 109 pp.

51. D'ALCANTARA Gonzague (15/12/83)
SERENA: a macroeconomic sectoral regional and national account econometric model for the Belgian economy. Leuven, KUL, 1983. 595 pp.

52. D'HAVE Piet (24/02/84)
De vraag naar geld in België. Leuven, KUL, 1984. XI, 318 pp.

53. MAES Ivo (16/03/84)
The contribution of J.R. Hicks to macro-economic and monetary theory. Leuven, KUL, 1984. V, 224 pp.

54. SUBIANTO Bambang (13/09/84)
A study of the effects of specific taxes and subsidies on a firms' R&D investment plan. s.l., s.n., 1984. V, 284 pp.

55. SLEUWAEGEN Leo (26/10/84)
Location and investment decisions by multinational enterprises in Belgium and Europe. Leuven, KUL, 1984. XII, 247 pp.

56. GEYSKENS Erik (27/03/85)
Produktietheorie en dualiteit. Leuven, s.n., 1985. VII, 392 pp.

57. COLE Frank (26/06/85)
Some algorithms for geometric programming. Leuven, KUL, 1985. 166 pp.

58. STANDAERT Stan (26/09/86)
A study in the economics of repressed consumption. Leuven, KUL, 1986. X, 380 pp.

59. DELBEKE Jos (03/11/86)
Trendperioden in de geldhoeveelheid van België 1877-1983: een theoretische en empirische analyse van de "Banking school" hypothese. Leuven, KUL, 1986. XII, 430 pp.

60. VANTHIENEN Jan (08/12/86)
Automatiseringsaspecten van de specificatie, constructie en manipulatie van beslissingstabellen. Leuven, s.n., 1986. XIV, 378 pp.

61. LUYTEN Robert (30/04/87)
A systems-based approach for multi-echelon production/inventory systems. s.l., s.n., 1987. 3V.

62. MERCKEN Roger (27/04/87)
De invloed van de data base benadering op de interne controle. Leuven, s.n., 1987. XIII, 346 pp.

63. VAN CAYSEELE Patrick (20/05/87)
Regulation and international innovative activities in the pharmaceutical industry. s.l., s.n., 1987. XI, 169 pp.

64. FRANCOIS Pierre (21/09/87)
De empirische relevantie van de independence from irrelevant alternatives. Assumptie indiscrete keuzemodellen. Leuven, s.n., 1987. IX, 379 pp.

65. DECOSTER André (23/09/88)
Family size, welfare and public policy. Leuven, KUL. Faculteit Economische en Toegepaste Economische Wetenschappen, 1988. XIII, 444 pp.

66. HEIJNEN Bart (09/09/88)
Risicowijziging onder invloed van vrijstellingen en herverzekeringen: een theoretische analyse van optimaliteit en premiebepaling. Leuven, KUL. Faculteit Economische en Toegepaste Economische Wetenschappen, 1988. onregelmatig gepagineerd.

67. GEEROMS Hans (14/10/88)
Belastingvermijding. Theoretische analyse van de determinanten van de belastingontduiking en de belastingontwijking met empirische verificaties. Leuven, s.n., 1988. XIII, 409, 5 pp.

68. PUT Ferdi (19/12/88)
Introducing dynamic and temporal aspects in a conceptual (database) schema. Leuven, KUL. Faculteit Economische en Toegepaste Economische Wetenschappen, 1988. XVIII, 415 pp.

69. VAN ROMPUY Guido (13/01/89)
A supply-side approach to tax reform programs. Theory and empirical evidence for Belgium. Leuven, KUL. Faculteit Economische en Toegepaste Economische Wetenschappen, 1989. XVI, 189, 6 pp.

70. PEETERS Ludo (19/06/89)
Een ruimtelijk evenwichtsmodel van de graanmarkten in de E.G.: empirische specificatie en beleidstoepassingen. Leuven, K.U.Leuven. Faculteit Economische en Toegepaste Economische Wetenschappen, 1989. XVI, 412 pp.

71. PACOLET Jozef (10/11/89)
Marktstructuur en operationele efficiëntie in de Belgische financiële sector. Leuven, K.U.Leuven. Faculteit Economische en Toegepaste Economische Wetenschappen, 1989. XXII, 547 pp.

72. VANDEBROEK Martina (13/12/89)
Optimalisatie van verzekeringscontracten en premieberekeningsprincipes. Leuven, K.U.Leuven. Faculteit Economische en Toegepaste Economische Wetenschappen, 1989. 95 pp.

73. WILLEKENS Francois (1990)
Determinance of government growth in industrialized countries with applications to Belgium. Leuven, K.U.Leuven. Faculteit Economische en Toegepaste Economische Wetenschappen, 1990. VI, 332 pp.

74. VEUGELERS Reinhilde (02/04/90)
Scope decisions of multinational enterprises. Leuven, K.U.Leuven. Faculteit Economische en Toegepaste Economische Wetenschappen, 1990. V, 221 pp.

75. KESTELOOT Katrien (18/06/90)
Essays on performance diagnosis and tacit cooperation in international oligopolies. Leuven, K.U.Leuven. Faculteit Economische en Toegepaste Economische Wetenschappen, 1990. 227 pp.

76. WU Changqi (23/10/90) Strategic aspects of oligopolistic vertical integration. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1990. VIII, 222 pp.

77. ZHANG Zhaoyong (08/07/91)
A disequilibrium model of China's foreign trade behaviour. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1991. XII, 256 pp.

78. DHAENE Jan (25/11/91)
Verdelingsfuncties, benaderingen en foutengrenzen van stochastische grootheden geassocieerd aan verzekeringspolissen en -portefeuilles. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1991. 146 pp.

79. BAUWELINCKX Thierry (07/01/92)
Hierarchical credibility techniques. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1992. 130 pp.

80. DEMEULEMEESTER Erik (23/3/92)
Optimal algorithms for various classes of multiple resource-constrained project scheduling problems. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1992. 180 pp.

81. STEENACKERS Anna (1/10/92)
Risk analysis with the classical actuarial risk model: theoretical extensions and applications to Reinsurance. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1992. 139 pp.

82. COCKX Bart (24/09/92)
The minimum income guarantee. Some views from a dynamic perspective. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1992. XVII, 401 pp.

83. MEYERMANS Eric (06/11/92)
Econometric allocation systems for the foreign exchange market: Specification, estimation and testing of transmission mechanisms under currency substitution. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1992. XVIII, 343 pp.

84. CHEN Guoqing (04/12/92)
Design of fuzzy relational databases based on fuzzy functional dependency. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1992. 176 pp.

85. CLAEYS Christel (18/02/93)
Vertical and horizontal category structures in consumer decision making: The nature of product hierarchies and the effect of brand typicality. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1993. 348 pp.

86. CHEN Shaoxiang (25/03/93)
The optimal monitoring policies for some stochastic and dynamic production processes. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1993. 170 pp.

87. OVERWEG Dirk (23/04/93)
Approximate parametric analysis and study of cost capacity management of computer configurations. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1993. 270 pp.

88. DEWACHTER Hans (22/06/93)
Nonlinearities in speculative prices: The existence and persistence of nonlinearity in foreign exchange rates. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1993. 151 pp.

89. LIN Liangqi (05/07/93)
Economic determinants of voluntary accounting choices for R & D expenditures in Belgium. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1993. 192 pp.

90. DHAENE Geert (09/07/93)
Encompassing: formulation, properties and testing. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1993. 117 pp.

91. LAGAE Wim (20/09/93)
Marktconforme verlichting van soevereine buitenlandse schuld door private crediteuren: een neo-institutionele analyse. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1993. 241 pp.

92. VAN DE GAER Dirk (27/09/93)
Equality of opportunity and investment in human capital. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1993. 172 pp.

93. SCHROYEN Alfred (28/02/94)
Essays on redistributive taxation when monitoring is costly. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1994. 203 pp. + V.

94. STEURS Geert (15/07/94)
Spillovers and cooperation in research and development. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1994. 266 pp.

95. BARAS Johan (15/09/94)
The small sample distribution of the Wald, Lagrange multiplier and likelihood ratio tests for homogeneity and symmetry in demand analysis: a Monte Carlo study. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1994. 169 pp.

96. GAEREMYNCK Ann (08/09/94)
The use of depreciation in accounting as a signalling device. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1994. 232 pp.

97. BETTENDORF Leon (22/09/94)
A dynamic applied general equilibrium model for a small open economy. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1994. 149 pp.

98. TEUNEN Marleen (10/11/94)
Evaluation of interest randomness in actuarial quantities. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1994. 214 pp.

99. VAN OOTEGEM Luc (17/01/95)
An economic theory of private donations. Leuven. K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1995. 236 pp.

100. DE SCHEPPER Ann (20/03/95)
Stochastic interest rates and the probabilistic behaviour of actuarial functions. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1995. 211 pp.

101. LAUWERS Luc (13/06/95)
Social choice with infinite populations. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1995. 79 pp.

102. WU Guang (27/06/95)
A systematic approach to object-oriented business modeling. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1995. 248 pp.

103. WU Xueping (21/08/95)
Term structures in the Belgian market: model estimation and pricing error analysis. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1995. 133 pp.

104. PEPERMANS Guido (30/08/95)
Four essays on retirement from the labor force. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1995. 128 pp.

105. ALGOED Koen (11/09/95)
Essays on insurance: a view from a dynamic perspective. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1995. 136 pp.

106. DEGRYSE Hans (10/10/95)
Essays on financial intermediation, product differentiation, and market structure. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1995. 218 pp.

107. MEIR Jos (05/12/95)
Het strategisch groepsconcept toegepast op de Belgische financiële sector. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1995. 257 pp.

108. WIJAYA Miryam Lilian (08/01/96)
Voluntary reciprocity as an informal social insurance mechanism: a game theoretic approach. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1996. 124 pp.

109. VANDAELE Nico (12/02/96)
The impact of lot sizing on queueing delays: multi product, multi machine models. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1996. 243 pp.

110. GIELENS Geert (27/02/96)
Some essays on discrete time target zones and their tails. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1996. 131 pp.

111. GUILLAUME Dominique (20/03/96)
Chaos, randomness and order in the foreign exchange markets. Essays on the modelling of the markets. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1996. 171 pp.

112. DEWIT Gerda (03/06/96)
Essays on export insurance subsidization. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1996. 186 pp.

113. VAN DEN ACKER Carine (08/07/96)
Belief-function theory and its application to the modeling of uncertainty in financial statement auditing. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1996. 147 pp.

114. IMAM Mahmood Osman (31/07/96)
Choice of IPO Flotation Methods in Belgium in an Asymmetric Information Framework and Pricing of IPO's in the Long-Run. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1996. 221 pp.

115. NICAISE Ides (06/09/96)
Poverty and Human Capital. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1996. 209 pp.

116. EYCKMANS Johan (18/09/97)
On the Incentives of Nations to Join International Environmental Agreements. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1997. XV + 348 pp.

117. CRISOLOGO-MENDOZA Lorelei (16/10/97)
Essays on Decision Making in Rural Households: a study of three villages in the Cordillera Region of the Philippines. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1997. 256 pp.

118. DE REYCK Bert (26/01/98)
Scheduling Projects with Generalized Precedence Relations: Exact and Heuristic Procedures. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1998. XXIV + 337 pp.

119. VANDEMAELE Sigrid (30/04/98)
Determinants of Issue Procedure Choice within the Context of the French IPO Market: Analysis within an Asymmetric Information Framework. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1998. 241 pp.

120. VERGAUWEN Filip (30/04/98)
Firm Efficiency and Compensation Schemes for the Management of Innovative Activities and Knowledge Transfers. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1998. VIII + 175 pp.

121. LEEMANS Herlinde (29/05/98)
The Two-Class Two-Server Queueing Model with Nonpreemptive Heterogeneous Priority Structures. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1998. 211 pp.

122. GEYSKENS Inge (4/09/98)
Trust, Satisfaction, and Equity in Marketing Channel Relationships. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1998. 202 pp.

123. SWEENEY John (19/10/98)
Why Hold a Job ? The Labour Market Choice of the Low-Skilled. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1998. 278 pp.

124. GOEDHUYS Micheline (17/03/99)
Industrial Organisation in Developing Countries, Evidence from Côte d'Ivoire. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1999. 251 pp.

125. POELS Geert (16/04/99)
On the Formal Aspects of the Measurement of Object-Oriented Software Specifications. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1999. 507 pp.

126. MAYERES Inge (25/05/99)
The Control of Transport Externalities: A General Equilibrium Analysis. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1999. XIV + 294 pp.

127. LEMAHIEU Wilfried (5/07/99)
Improved Navigation and Maintenance through an Object-Oriented Approach to Hypermedia Modelling. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1999. 284 pp.

128. VAN PUYENBROECK Tom (8/07/99)
Informational Aspects of Fiscal Federalism. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1999. 192 pp.

129. VAN DEN POEL Dirk (5/08/99)
Response Modeling for Database Marketing Using Binary Classification. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1999. 342 pp.

130. GIELENS Katrijn (27/08/99)
International Entry Decisions in the Retailing Industry: Antecedents and Performance Consequences. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1999. 336 pp.

131. PEETERS Anneleen (16/12/99)
Labour Turnover Costs, Employment and Temporary Work. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1999. 207 pp.

132. VANHOENACKER Jurgen (17/12/99)
Formalizing a Knowledge Management Architecture Meta-Model for Integrated Business Process Management. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1999. 252 pp.

133. NUNES Paulo (20/03/2000)
Contingent Valuation of the Benefits of Natural Areas and its Warmglow Component. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2000. XXI + 282 pp.

134. VAN DEN CRUYCE Bart (7/04/2000)
Statistische discriminatie van allochtonen op jobmarkten met rigide lonen. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2000. XXIII + 441 pp.

135. REPKINE Alexandre (15/03/2000)
Industrial restructuring in countries of Central and Eastern Europe: Combining branch-, firm- and product-level data for a better understanding of Enterprises' behaviour during transition towards market economy. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2000. VI + 147 pp.

136. AKSOY, Yunus (21/06/2000)
Essays on international price rigidities and exchange rates. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2000. IX + 236 pp.

137. RIYANTO, Yohanes Eko (22/06/2000)
Essays on the internal and external delegation of authority in firms. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2000. VIII + 280 pp.

138. HUYGHEBAERT, Nancy (20/12/2000)
The Capital Structure of Business Start-ups. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2000. VIII + 332 pp.

139. FRANCKX Laurent (22/01/2001)
Ambient Inspections and Commitment in Environmental Enforcement. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. VIII + 286 pp.

140. VANDILLE Guy (16/02/2001)
Essays on the Impact of Income Redistribution on Trade. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. VIII + 176 pp.

141. MARQUERING Wessel (27/04/2001)
Modeling and Forecasting Stock Market Returns and Volatility. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. V + 267 pp.

142. FAGGIO Giulia (07/05/2001)
Labor Market Adjustment and Enterprise Behavior in Transition. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. 150 pp.

143. GOOS Peter (30/05/2001)
The Optimal Design of Blocked and Split-plot experiments. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. X + 224 pp.

144. LABRO Eva (01/06/2001)
Total Cost of Ownership Supplier Selection based on Activity Based Costing and Mathematical Programming. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. 217 pp.

145. VANHOUCKE Mario (07/06/2001)
Exact Algorithms for various Types of Project Scheduling Problems. Nonregular Objectives and time/cost Trade-offs. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. 316 pp.

146. BILSEN Valentijn (28/08/2001)
Entrepreneurship and Private Sector Development in Central European Transition Countries. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. XVI + 188 pp.

147. NIJS Vincent (10/08/2001)
Essays on the dynamic Category-level Impact of Price promotions. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001.

148. CHERCHYE Laurens (24/09/2001)
Topics in Non-parametric Production and Efficiency Analysis. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. VII + 169 pp.

149. VAN DENDER Kurt (15/10/2001)
Aspects of Congestion Pricing for Urban Transport. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. VII + 203 pp.

150. CAPEAU Bart (26/10/2001)
In defence of the excess demand approach to poor peasants' economic behaviour. Theory and Empirics of non-recursive agricultural household modelling. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. XIII + 286 pp.

151. CALTHROP Edward (09/11/2001)
Essays in urban transport economics. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001.

152. VANDER BAUWHEDE Heidi (03/12/2001)
Earnings management in an Non-Anglo-Saxon environment. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. 408 pp.

153. DE BACKER Koenraad (22/01/2002)
Multinational firms and industry dynamics in host countries : the case of Belgium. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. VII + 165 pp.

154. BOUWEN Jan (08/02/2002)
Transactive memory in operational workgroups. Concept elaboration and case study. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. 319 pp. + appendix 102 pp.

155. VAN DEN BRANDE Inge (13/03/2002)
The psychological contract between employer and employee : a survey among Flemish employees. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. VIII + 470 pp.

156. VEESTRAETEN Dirk (19/04/2002)
Asset Price Dynamics under Announced Policy Switching. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. 176 pp.

157. PEETERS Marc (16/05/2002)
One Dimensional Cutting and Packing : New Problems and Algorithms. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. IX + 247 pp.

158. SKUDELNY Frauke (21/05/2002)
Essays on The Economic Consequences of the European Monetary Union. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002.

159. DE WEERDT Joachim (07/06/2002)
Social Networks, Transfers and Insurance in Developing countries. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. VI + 129 pp.

160. TACK Lieven (25/06/2002)
Optimal Run Orders in Design of Experiments. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. XXXI + 344 pp.

161. POELMANS Stephan (10/07/2002)
Making Workflow Systems work. An investigation into the Importance of Task-appropriation fit, End-user Support and other Technological Characteristics. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. 237 pp.

162. JANS Raf (26/09/2002)
Capacitated Lot Sizing Problems : New Applications, Formulations and Algorithms. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002.

163. VIAENE Stijn (25/10/2002)
Learning to Detect Fraud from enriched Insurance Claims Data (Context, Theory and Applications). Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. 315 pp.

164. AYALEW Tekabe (08/11/2002)
Inequality and Capital Investment in a Subsistence Economy.Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. V + 148 pp.

165. MUES Christophe (12/11/2002)
On the Use of Decision Tables and Diagrams in Knowledge Modeling and Verification. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. 222 pp.

166. BROCK Ellen (13/03/2003)
The Impact of International Trade on European Labour Markets. K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002.

167. VERMEULEN Frederic (29/11/2002)
Essays on the collective Approach to Household Labour Supply. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. XIV + 203 pp.

168. CLUDTS Stephan (11/12/2002)
Combining participation in decision-making with financial participation : theoretical and empirical perspectives. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. XIV + 247 pp.

169. WARZYNSKI Frederic (09/01/2003)
The dynamic effect of competition on price cost margins and innovation. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

170. VERWIMP Philip (14/01/2003)
Development and genocide in Rwanda ; a political economy analysis of peasants and power under the Habyarimana regime. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

171. BIGANO Andrea (25/02/2003)
Environmental regulation of the electricity sector in a European Market Framework. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003. XX + 310 pp.

172. MAES Konstantijn (24/03/2003)
Modeling the Term Structure of Interest Rates Across Countries. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003. V+246 pp.

173. VINAIMONT Tom (26/02/2003)
The performance of One- versus Two-Factor Models of the Term Structure of Interest Rates. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen. 2003.

174. OOGHE Erwin (15/04/2003)
Essays in multi-dimensional social choice. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003. VIII+108 pp.

175. FORRIER Anneleen (25/04/2003)
Temporary employment, employability and training. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

176. CARDINAELS Eddy (28/04/2003)
The role of cost system accuracy in managerial decision making. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003. 144 pp.

177. DE GOEIJ Peter (02/07/2003)
Modeling Time-Varying Volatility and Interest Rates. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003. VII+225 pp.

178. LEUS Roel (19/09/2003)
The generation of stable project plans. Complexity and exact algorithms. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

179. MARINHEIRO Carlos (23/09/2003)
EMU and fiscal stabilisation policy : the case of small countries. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

180. BAESSENS Bart (24/09/2003)
Developing intelligent systems for credit scoring using machine learning techniques. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

181. KOCZY Laszlo (18/09/2003)
Solution concepts and outsider behaviour in coalition formation games. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

182. ALTOMONTE Carlo (25/09/2003)
Essays on Foreign Direct Investment in transition countries : learning from the evidence. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

183. DRIES Liesbeth (10/11/2003)
Transition, Globalisation and Sectoral Restructuring: Theory and Evidence from the Polish Agri-Food Sector. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

184. DEVOOGHT Kurt (18/11/2003)
Essays On Responsibility-Sensitive Egalitarianism and the Measurement of Income Inequality. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

185. DELEERSNYDER Barbara (28/11/2003)
Marketing in Turbulent Times. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

186. ALI Daniel (19/12/2003)
Essays on Household Consumption and Production Decisions under Uncertainty in Rural Ethiopia. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

187. WILLEMS Bert (14/01/2004)
Electricity networks and generation market power. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

188. JANSSENS Gust (30/01/2004)
Advanced Modelling of Conditional Volatility and Correlation in Financial Markets. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

189. THOEN Vincent (19/01/2004)
On the valuation and disclosure practices implemented by venture capital providers. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

190. MARTENS Jurgen (16/02/2004)
A fuzzy set and stochastic system theoretic technique to validate simulation models. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

191. ALTAVILLA Carlo (21/05/2004)
Monetary policy implementation and transmission mechanisms in the Euro area. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

192. DE BRUYNE Karolien (07/06/2004)
Essays in the location of economic activity. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

193. ADEM Jan (25/06/2004)
Mathematical programming approaches for the supervised classification problem. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

194. LEROUGE Davy (08/07/2004)
Predicting Product Preferences : the effect of internal and external cues. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

195. VANDENBROECK Katleen (16/07/2004)
Essays on output growth, social learning and land allocation in agriculture : micro-evidence from Ethiopia and Tanzania. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

196. GRIMALDI Maria (03/09/2004)
The exchange rate, heterogeneity of agents and bounded rationality. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

197. SMEDTS Kristien (26/10/2004)
Financial integration in EMU in the framework of the no-arbitrage theory. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

198. KOEVOETS Wim (12/11/2004)
Essays on Unions, Wages and Employment. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

199. CALLENS Marc (22/11/2004)
Essays on multilevel logistic Regression. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

200. RUGGOO Arvind (13/12/2004)
Two stage designs robust to model uncertainty. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

201. HOORELBEKE Dirk (28/01/2005)
Bootstrap and Pivoting Techniques for Testing Multiple Hypotheses. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

202. ROUSSEAU Sandra (17/02/2005)
Selecting Environmental Policy Instruments in the Presence of Incomplete Compiance. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

203. VAN DER MEULEN Sofie (17/02/2005)
Quality of Financial Statements : Impact of the external auditor and applied accounting standards. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

204. DIMOVA Ralitza (21/02/2005)
Winners and Losers during Structural Reform and Crisis : the Bulgarian Labour Market Perspective. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

205. DARKIEWICZ Grzegorz (28/02/2005)
Value-at-risk in Insurance and Finance : the Comonotonicity Approach. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

206. DE MOOR Lieven (20/05/2005)
The Structure of International Stock Returns : Size, Country and Sector Effects in Capital Asset Pricing. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

207. EVERAERT Greetje (27/06/2005)
Soft Budget Constraints and Trade Policies : The Role of Institutional and External Constraints. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

208. SIMON Steven (06/07/2005)
The Modeling and Valuation of complex Derivatives : The Impact of the Choice of the term structure model. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

209. MOONEN Linda (23/09/2005)
Algorithms for some Graph-Theoretical Optimization Problems. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

210. COUCKE Kristien (21/09/2005)
Firm and industry adjustment under de-industrialisation and globalization of the Belgian economy. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

211. DECAMPS Marc (21/10/2005)
Some actuarial and financial applications of generalized diffusion processes. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

212. KIM Helena (29/11/2005)
Escalation games: an instrument to analyze conflicts. The strategic approach to the bargaining problem. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

213. GERMENJI Etleva (06/01/2006)
Essays on the Economics of Emigration from Albania. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

214. BELIEN Jeroen (18/01/2006)
Exact and Heuristic Methodologies for Scheduling in Hospitals: Problems, Formulations and Algorithms. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

215. JOOSSENS Kristel (10/02/2006)
Robust discriminant analysis. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

216. VRANKEN Liesbet (13/02/2006)
Land markets and production efficiency in transition economies. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

217. VANSTEENKISTE Isabel (22/02/2006)
Essays on non-linear modelling in international macroeconomics. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

218. WUYTS Gunther (31/03/2006)
Essays on the liquidity of financial markets. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

219. DE BLANDER Rembert (28/04/2006)
Essays on endogeneity and parameter heterogeneity in cross-section and panel data. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

220. DE LOECKER Jan (12/05/2006)
Industry dynamics and productivity. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

221. LEMMENS Aurélie (12/05/2006)
Advanced classification and time-series methods in marketing. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

222. VERPOORTEN Marijke (22/05/2006)
Conflict and survival: an analysis of shocks, coping strategies and economic mobility in Rwanda, 1990-2002. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

223. BOSMANS Kristof (26/05/2006)
Measuring economic inequality and inequality aversion. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

224. BRENKERS Randy (29/05/2006)
Policy reform in a market with differentiated products: applications from the car market. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

225. BRUYNEEL Sabrina (02/06/2006)
Self-control depletion: Mechanisms and its effects on consumer behavior. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

226. FAEMS Dries (09/06/2006)
Collaboration for innovation: Processes of governance and learning. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

227. BRIERS Barbara (28/06/2006)
Countering the scrooge in each of us: on the marketing of cooperative behavior. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

228. ZANONI Patrizia (04/07/2006)
Beyond demography: Essays on diversity in organizations. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

229. VAN DEN ABBEELE Alexandra (11/09/2006)
Management control of interfirm relations: the role of information. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

230. DEWAELHEYNS Nico (18/09/2006)
Essays on internal capital markets, bankruptcy and bankruptcy reform. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

231. RINALDI Laura (19/09/2006)
Essays on card payments and household debt. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

232. DUTORDOIR Marie (22/09/2006)
Determinants and stock price effects of Western European convertible debt offerings: an empirical analysis. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

233. LYKOGIANNI Elissavet (20/09/2006)
Essays on strategic decisions of multinational enterprises: R&D decentralization, technology transfers and modes of foreign entry. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

234. ZOU Jianglei (03/10/2006)
Inter-firm ties, plant networks, and multinational firms: essays on FDI and trade by Japanse firms. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

235. GEYSKENS Kelly (12/10/2006)
The ironic effects of food temptations on self-control performance. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

236. BRUYNSEELS Liesbeth (17/10/2006)
Client strategic actions, going-concern audit opinions and audit reporting errors. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

237. KESSELS Roselinde (23/10/2006)
Optimal designs for the measurement of consumer preferences. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

238. HUTCHINSON John (25/10/2006)
The size distribution and growth of firms in transition countries. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

239. RENDERS Annelies (26/10/2006)
Corporate governance in Europe: The relation with accounting standards choice, performance and benefits of control. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

240. DE WINNE Sophie (30/10/2006)
Exploring terra incognita: human resource management and firm performance in small and medium-sized businesses. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

241. KADITI Eleni (10/11/2006)
Foreign direct investments in transition economies. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

242. ANDRIES Petra (17/11/2006)
Technology-based ventures in emerging industries: the quest for a viable business model. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

243. BOUTE Robert (04/12/2006)
The impact of replenishment rules with endogenous lead times on supply chain performance. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

244. MAES Johan (20/12/2006)
Corporate entrepreneurship: an integrative analysis of a resource-based model. Evidence from Flemish enterprises. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

245. GOOSSENS Dries (20/12/2006)
Exact methods for combinatorial auctions. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

246. GOETHALS Frank (22/12/2006)
Classifying and assessing extended enterprise integration approaches. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

247. VAN DE VONDER Stijn (22/12/2006)
Proactive-reactive procedures for robust project scheduling. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.