Universiteit Gent
Faculteit Economie en Bedrijfskunde
Vakgroep Beleidsinformatica en Operationeel Beheer

Multi-Mode Resource-Constrained Project Scheduling Problem

Metaheuristic Solution Procedures and Extensions

Vincent Van Peteghem

Proefschrift tot het bekomen van de graad van
Doctor in de Toegepaste Economische Wetenschappen: Handelsingenieur
Academiejaar 2009-2010

Universiteit Gent
Faculteit Economie en Bedrijfskunde
Vakgroep Beleidsinformatica en Operationeel Beheer

Promotor
Prof. dr. Mario Vanhoucke

Doctoral jury
Prof. dr. Marc De Clercq - Dean
Prof. dr. Patrick Van Kenhove - Academic Secretary
Prof. dr. Erik Demeulemeester - Katholieke Universiteit Leuven
Prof. dr. Luc Chalmet - Universiteit Gent/Universiteit Antwerpen
Prof. dr. Rainer Kolisch - Technische Universität München - Germany
Prof. dr. Francisco Ballestin - Public University of Navarra - Spain

Universiteit Gent
Faculteit Economie en Bedrijfskunde

Vakgroep Beleidsinformatica en Operationeel Beheer
Tweekerkenstraat 2
B-9000 Gent, België

Tel.: +32-9-264.35.17
Fax.: +32-9-264.42.86

Proefschrift tot het bekomen van de graad van
Doctor in de Toegepaste Economische Wetenschappen: Handelsingenieur
Academiejaar 2009-2010

Dankwoord

Our greatest weakness lies in giving up. The most certain way to succeed is always to try just one more time.

Thomas Alva Edison

Hier ligt het dan. Mijn doctoraat. Mijn dagen schrijven van papers. Mijn weken zoeken naar dat halve procentje verbetering. Mijn maanden van ploeteren in code. En toch, nu alles neergeschreven, gelezen en herlezen is en het geheel netjes is ingebonden, blijven vooral tientallen herinneringen over die niets - of slechts van veraf - met het onderwerp van mijn doctoraat te maken hebben. De Brug. MISTA. Ierland. Mails. Personeelscompetitie. Vlerick. Verlofiche. 12urenloop. Schriftje. Seminariewerk. Deadlines. Apple. Restaurant zoeken. Assistants4Life. Poitiers. Lesgeven. Weer Poitiers. Tübingen. Nog ééntje. Fiona. Bowlingkampioenschap. 9 mei 2007. Stukje taart. In order to. Tours. En vooral: nooit opgeven.

Toch zou dit doctoraat - en alle bijhorende herinneringen - er nooit geweest zijn zonder de hulp van velen. Met dit dankwoord hoop ik dan ook allen die betrokken zijn geweest bij dit doctoraat te bedanken.

Vooreerst een heel bijzonder woord van dank aan mijn promotor, prof. dr. Mario Vanhoucke. Mario, het was voor mij een ongelofelijke eer om de afgelopen jaren samen te werken met jou. Je deur stond altijd open om iets te vragen en vele uren hebben we samen gespendeerd aan je ronde tafel. Je nooit aflatende enthousiasme voor onderzoek trok me telkens opnieuw mee in die - toch wel - wondere wereld van project management. We hebben even moeten zoeken hoe we mijn onderzoek moesten aanpakken, maar telkens hebben we een nieuwe poging ondernomen en doorgezet. Het gesprek dat we in Tübingen hebben gehad heeft er ongetwijfeld voor gezorgd dat dit document hier vandaag ligt. Mario, bedankt voor je steun, je hulp en de kansen die ik kreeg.

Ook een woord van dank aan prof. dr. Luc Chalmet. Luc, de afgelopen jaren hebben we intensief samengewerkt aan de cursus Productie- en Logistiek beleid. We hebben dit steeds aangepakt in onderlinge samenspraak en steeds kreeg ik hierbij van jou alle kansen. Het deed me dan ook enorm veel plezier toen je lid wou worden van mijn doctoraatsjury. Luc, bedankt voor de vlotte en aangename samenwerking de afgelopen jaren.

Ook een woord van dank aan de andere leden van mijn examenjury, prof. dr. Erik Demeulemeester, prof. dr. Rainer Kolisch en prof. dr. Francisco Ballestin. Erik, net zoals vele anderen hier op onze onderzoeksgroep heb ik de basisknepen

van operationeel onderzoek meegekregen tijdens je lessen in Leuven. Ze hebben mee de basis gelegd voor dit doctoraat. Bedankt voor alle opmerkingen, feedback en ondersteuning. Rainer, you are one of the persons cited the most in my dissertation, so it meant a lot to me that you became a member of my doctoral jury. Moreover, your very detailed and elaborate list of feedback significantly improved the quality of my work. My sincere thanks to you. Francisco, thank you for critically revising my dissertation and for your insightful comments and suggestions to increase the quality of this work.

Ook nog een woord van dank aan prof. dr. Johan Christiaens voor de kans die hij mij vele jaren geleden geboden heeft om als assistent te starten aan deze faculteit. Ondanks de korte periode dat we samengewerkt hebben, hou ik vele fijne herinneringen over aan onze samenwerking.

Een stimulerende onderzoeksomgeving begint natuurlijk bij een goede werkomgeving. En daarom vooreerst een woord van dank aan de Universiteit Gent. Meer dan tien jaar na mijn eerste les ben ik nog steeds ongelofelijk trots om aan deze instelling te mogen studeren en werken. Het is voor mij in ieder geval een grote eer geweest om deel uit te maken van deze universiteitsgemeenschap en ik ben blij dat ik met dit doctoraat een beperkte bijdrage heb kunnen leveren aan de verdere uitbouw van onze universiteit.

Graag wil ik dan ook alle collega's aan onze vakgroep en faculteit van harte bedanken voor de vele fijne en ontspannende momenten. Een speciaal woord van dank aan mijn bureaugenoten Christophe, Jeroen, Thomas en Veronique. Hoewel we soms de 'stille bureau' genoemd worden, zorgden onze korte gesprekken steeds voor de nodige afleiding. Ook mijn oude bureaugenoten en collega's bij de vakgroep accountancy wil ik graag bedanken voor de leuke tijd. Een speciaal woord van dank ook aan mijn (ex-)collega's Peter, Arne, Dieter, Frederik en Broos. Als ik het even niet meer zag zitten, eens goed wou klagen of gewoon zin had in een gesprek of een grap, kon ik steeds bij jullie terecht. Bedankt. Ook een heel groot woord van dank aan het ondersteunend personeel: de mensen van het secretariaat (Martine en Ann), het decanaat en het onderhoudspersoneel.

Dit doctoraat zou er ook niet gekomen zijn door (af en toe) te ontspannen. Eén van mijn meest dierbare ontspanningen is ongetwijfeld De Pinte Leeft! Peter, Lieven, Matthijs, Guy, Dieter en Peter, samen hebben we de afgelopen jaren heel wat uit de grond gestampt. Ik hoop dat we ook in de toekomst onze doelstelling kunnen blijven waarmaken, maar met een Pintenaer in de hand zal dit ongetwijfeld lukken. Ook mijn collega-volleyballers wil ik bedanken. Aangezien onderzoek dikwijls eenzaam is, heb ik altijd enorm genoten van de sfeer in onze ploeg, van de punten tijdens de wedstrijd en de pinten erna...

Daarnaast zijn er een aantal mensen die ik heel bijzonder wil bedanken voor hun onvoorwaardelijke steun de afgelopen jaren. Als ik belde, hen tegenkwam of bij hen langsging, kon ik steeds alles tegen hen kwijt. Dus daarom een hele grote dank-je-wel aan Matthijs, Wim, Jeroen, Helen, Nele, Peter en Stefaan.

Ook een bijzonder woord van dank aan Philippe en Luce. Het was leuk om op elk moment welkom te zijn. Steeds mocht ik mee aanschuiven aan tafel en leefden jullie mee met de vorderingen van mijn doctoraat. Bedankt!

Ook mijn familie wil ik danken voor de steun en in het bijzonder mijn grootouders. Grootouders zijn met hun financiële beloningssysteem dikwijls de grootste motivator voor goede studieprestaties, maar dat waren hun interesse en meeleven evenzeer. De strenge, maar goedkeurende blik van pepe, de korte bezoekjes bij oma net voor ik naar een examen vertrok en de steeds terugkerende vragen van Bobo, elk zorgden ze voor de nodige stimulans. Ook mijn allerliefste zusjes mogen niet ontbreken in dit dankwoord. Julie en Elisa, hoewel ik mij het waarschijnlijk nog vaak zal beklagen, ik kan alleen maar zeggen dat ik mij geen betere zussen kan indenken. Steeds stonden jullie klaar om mij te helpen, af en toe met volle goesting, maar ongetwijfeld vaak tegen jullie zin. Toch kon ik steeds op jullie rekenen. Bedankt Jules en Lizie, ook voor het nalezen van dit doctoraat.

Tot slot wil ik zeker mijn papa en mama bedanken. Ik was waarschijnlijk niet altijd de gemakkelijkste. Ik vertelde nooit waar ik mee bezig was of gaf enkel mijn standaardantwoord ’een beetje’ als jullie vroegen ’vlot het een beetje?’. Toch hoop ik dat dit doctoraat ook voor jullie een bekroning is voor de manier waarop jullie me al die jaren steunden en ondersteunden. Ik ben jullie in ieder geval ontzettend dankbaar voor alles.

Save the best for last! Evelyn, wie had dit ooit gedacht toen we elkaar iets meer dan drie jaar geleden voor de eerste keer kruisten. We hebben sindsdien samen al heel wat beleefd, genoten en afgelachen. Je steun heeft me de afgelopen maanden geholpen om te blijven doorgaan. Bedankt om er steeds te zijn voor mij. Laten we elkaar nog lang gelukkig maken!

Gent, 28 mei 2010
Vincent Van Peteghem

Table of Contents

List of Figures

List of Tables

List of Acronyms

A

AIS	Artificial Immune System
AL	Activity list

B

BPGA	Bi-Population Genetic Algorithm

C

CNC	Coefficient of Network Complexity
CP	Critical Path
CPIM	Critical Path Improvement Method
CPU	Computer Processing Unit
CSLB	Critical Sequence Lower Bound

D

DTRTP	Discrete Time/Resource Tradeoff Problem

E

ERR	Excess of Resource Request

F

FIM	Feasibility Improvement Method

G

GA	Genetic Algorithm

L

LB	Lower Bound
LFT	Latest Finish Time
LST	Latest Start Time

M

MAP	Mode Assignment Problem
MASSP	Medium-term Audit-staff Scheduling Problem
ML	Mode List
MMLIB	Multi-mode Library
MRCPSP	Multi-mode Resource-constrained Project Scheduling Problem
MRCPSP/R	Multi-mode Resource-constrained Project Scheduling Problem with Renewable resources
MV	Mode Vector

N

NP	Non-deterministic Polynomial-time

O

OS	Order Strength

P

PMBOK	Project Management Body Of Knowledge
PS	Particle Swarm
PSPLIB	Project scheduling problem library
P-MRCPSP	Preempted Multi-mode Resource-Constrained Project Scheduling Problem

R

R	Renewable resources
RC	Resource constrainedness
RCPSP	Resource-constrained Project Scheduling Problem
RF	Resource Factor
RK	Random key
RNR	Renewable and Nonrenewable resources
RS	Resource Strength
RWK	Remaining Work Content

S

SA	Simulated Annealing
SGS	Schedule Generation Scheme
SLK	Slack
SS	Scatter search
SUM	Sum of durations

T

TO	Topological ordering

TS	Tabu Search
TWC	Total Work Content

W

WC	Work Content
WCIM	Work Content Improvement Method

Nederlandse samenvatting

Operations research (OR) heeft als doel processen binnen organisaties te verbeteren of te optimaliseren met behulp van hiervoor ontwikkelde technieken en modellen. De discipline kende zijn oorsprong tijdens WOII, toen aan de hand van wiskundige modellen de logistieke bevoorrading van militair materiaal en goederen werd gepland. In de jaren na de oorlog ontwikkelde OR zich ten volle en tot op vandaag worden technieken en procedures ontwikkeld om complexe problemen in de bedrijfswereld, de maatschappij en de industrie te analyseren en te optimaliseren.

Eén van de onderzoeksdomeinen waarbinnen OR actief is, is project management. Project management kan omschreven worden als het geheel van kennis, vaardigheden, tools en technieken om een project zo te plannen dat het aan alle projecteisen voldoet. Een project kan gedefinieerd worden als een tijdelijke inspanning met als doel het creëren van een uniek product of een unieke service (PMBOK). De bouw van piramides in Egypte, de ontwikkeling van een iPhone-applicatie, het schrijven van een doctoraat, de organisatie van een verkiezingscampagne of het bouwen van een huis zijn allemaal typische voorbeelden van projecten.

De voorbije jaren is het belang van project management enorm toegenomen. Tientallen boeken over project management zijn verschenen en project software pakketten zijn ontwikkeld of uitgebreid met nieuwe planningsmogelijkheden. Bovendien zijn verschillende planningsproblemen reeds uitvoerig bestudeerd in de academische literatuur en zijn talloze exacte, heuristische of metaheuristische oplossingsmethodes voorgesteld.

Eén van die planningsproblemen is het zogenaamde 'multi-mode resource-constrained project scheduling probleem', waarbij getracht wordt een project in een zo kort mogelijke duurtijd te plannen, rekening houdend met de volgorderelaties tussen de verschillende activiteiten én met de beschikbare hernieuwbare en niet-hernieuwbare middelen. Voor elk van de activiteiten zijn er bovendien meerdere uitvoeringsmogelijkheden.

Dit doctoraat is opgedeeld in twee delen. In een eerste deel worden drie metaheuristische oplossingsprocedures en een nieuwe dataset voorgesteld, terwijl in het tweede deel verschillende meer praktische concepten worden geïntroduceerd. Dit werk wordt afgesloten met een algemene conclusie en enkele suggesties voor verder onderzoek.

Deel I van dit doctoraat start met een introductie van het multi-mode resource-constrained project scheduling probleem en een overzicht van de beschikbare li-

teratuur. Aan de hand van een voorbeeld worden enkele veelgebruikte termen in de project planning literatuur voorgesteld. Vervolgens worden drie oplossingsmethodes ontwikkeld: een genetisch algoritme (GA), een artificieel immuun systeem algoritme (AIS) en een scatter search algoritme (SS).

Het voorgestelde genetisch algoritme verschilt van andere genetische oplossingsmethodes aangezien het gebruik maakt van twee populaties, één met left-justified schedules (waarbij alle activiteiten zo vroeg mogelijk gepland worden) en één met right-justified schedules (waarbij alle activiteiten zo laat mogelijk gepland worden). Het algoritme maakt ook gebruik van een generatieschema dat is uitgebreid met een methode die de gekozen mode van een activiteit tracht te optimaliseren door te kiezen voor de mode die resulteert in de laagst mogelijke eindtijd voor die activiteit.

De AIS procedure is gebaseerd op de principes van het menselijke immuun systeem. Wanneer ziektekiemen het menselijke lichaam binnendringen zullen de antigenen die in staat zijn om de ziektekiemen te bestrijden, zich vermenigvuldigen om op die manier zo snel mogelijk de ziekte te doen verdwijnen. Ditzelfde principe wordt toegepast in deze oplossingsmethode, die bovendien een procedure bevat om op een gecontroleerde manier de initiële populatie te genereren. Deze procedure is gebaseerd op experimentele resultaten die een link aantonen tussen bepaalde eigenschappen van de gekozen modes en de uiteindelijke duurtijd van het project.

Een laatste algoritme is een scatter search algoritme. Deze procedure maakt gebruik van verschillende verbeteringsmethodes die elk aangepast zijn aan de specifieke eigenschappen van de verschillende hernieuwbare en niet-hernieuwbare middelen. Aan de hand van parameters die de beperktheid van de middelen aangeeft, wordt de procedure gestuurd in de richting van de meest efficiënte verbeteringsmethode en op die manier wordt een zo optimaal mogelijke oplossing gezocht.

Elk van de voorgestelde procedures behaalde uitstekende resultaten op de bestaande benchmark datasets. Deze sets vertonen evenwel enkele beperkingen gezien de huidige evolutie in de ontwikkeling van metaheuristische oplossingsmethodes. Om die reden werd een nieuwe, verbeterde dataset ontwikkeld, die onderzoekers in staat moet stellen om hun oplossingen te vergelijken met andere procedures.

Om een vergelijking te kunnen maken tussen alle bestaande oplossingsmethodes hebben we elk algoritme dat beschikbaar is in de literatuur gecodeerd en getest op de bestaande en nieuwe datasets. Door alle testen uit te voeren op eenzelfde computer en met eenzelfde stopcriterium zijn we in staat geweest een duidelijke en faire vergelijking te maken. Onze voorgestelde algoritmes presteren bovendien uitstekend.

In het tweede deel van dit doctoraat worden een aantal uitbreidingen onder de loep genomen. Zo wordt in het eerste hoofdstuk van dit tweede deel de invloed nagegaan van het onderbreken van activiteiten: activiteiten kunnen dan op elke tijdstip stopgezet worden om later, zonder bijkomende kost, herstart te worden. De introductie van deze uitbreiding leidt tot een significante daling van de gemiddelde duurtijd van een project vergeleken met de situatie waarin geen onderbrekingen toegelaten worden.

Een andere uitbreiding is de introductie van leereffecten in een projectomgeving. Hierbij wordt verondersteld dat een persoon efficiënter wordt naarmate hij of zij langer aan een activiteit werkt. Dit leerconcept wordt vanuit drie verschillende hoeken bekeken. Ten eerste wordt nagegaan wat de invloed is van de introductie van het leerconcept op de totale duurtijd van een project en worden de verschillende parameters die hierop een invloed hebben geanalyseerd. Ten tweede bekijken we welke foutenmarge er moet aangenomen worden wanneer men geen rekening houdt met het leerconcept en tot slot achterhalen we dat door het tijdig incorporeren van de leereffecten significante verbeteringen kunnen gerealiseerd worden.

In het laatste deel van dit doctoraat wordt het genetisch algoritme uit deel I gebruikt om de planning van een audit kantoor te optimaliseren. In deze planning dienen audit teams toegewezen te worden aan verschillende audit taken. Er kan duidelijk aangetoond worden dat met het gebruik van optimalisatietechnieken significante verbeteringen kunnen gemaakt worden in de planning van de audit teams.

De bijdrage van dit doctoraat is drievoudig. Ten eerste werden drie state-of-the-art algoritmes gepresenteerd die in staat zijn om het multi-mode resource-constrained project scheduling probleem op een heel efficiënte manier op te lossen. Bovendien werd telkens specifieke project informatie gebruikt om de efficiëntie van de procedure te verhogen. Ten tweede werden verschillende stappen ondernomen om dit probleem uit te breiden naar meer realistische planningsproblemen. Het toelaten van het onderbreken van activiteiten en de introductie van leereffecten leidden tot nieuwe inzichten in het onderzoek van project planning. Tot slot worden met de ontwikkeling van een nieuwe dataset onderzoekers aangemoedigd om hun resultaten te vergelijken met die van andere procedures. Met deze nieuwe dataset is tevens de basis gelegd voor verder onderzoek van dit interessante planningsprobleem.

English summary

In the literature, a project is often described as a temporary endeavor undertaken to create a unique product or service. The management of those projects is accomplished through the use of the process of initiating, planning, executing, controlling and closing (PMBOK). To guide projects to success, it is important to construct feasible and cost-efficient schedules. Different research fields within project scheduling have been explored during recent years and a large set of exact, heuristic and metaheuristic solution procedures have been designed in order to tackle a wide variety of project scheduling problems.

In this work, we study the multi-mode resource-constrained project scheduling problem, which is a generalization of the resource-constrained project scheduling problem. Each activity in the project can be performed in different sets of modes, with specific activity durations and resource requirements. The objective of the MRCPSP is to find a mode and a start time for each activity such that the makespan is minimized and the schedule is feasible with respect to the precedence and renewable and nonrenewable resource constraints.

The work consists of two main parts. In the first part, three metaheuristic solution procedures are presented and a new benchmark dataset is proposed. In the second part, several practical concepts are introduced in order to take steps towards real-life scheduling problems. This work is concluded with an overall conclusion and several suggestions for further research.

Part I starts with an introduction of the problem under study. An overview of the available metaheuristic solution procedures is given and several concepts and definitions used in this work are explained and illustrated with an example. Furthermore, three new metaheuristic solution procedures are presented, i.e. a genetic algorithm, an artificial immune system and a scatter search procedure.

The genetic algorithm makes use of two separate populations and extends the serial schedule generation scheme by introducing a mode improvement procedure. This procedure improves the mode selection by choosing the feasible mode of a certain activity that minimizes the finish time of the activity. The artificial immune system algorithm makes use of mechanisms inspired by the vertebrate immune system, such as hypermutation and proliferation. A controlled mode assignment procedure is set up in order to generate the initial population. This procedure is based on experimental results which reveal a link between predefined mode list characteristics and the project makespan. Finally, the scatter search is executed with different improvement methods, each tailored to the specific characteristics of different renewable and nonrenewable resource scarceness values. These resource parameters have been introduced in project scheduling literature to measure a project instance's resource scarceness and are incorporated in the search process of the scatter search procedure.

All procedures proved to be very successful on the currently available benchmark datasets, the PSPLIB dataset (Kolisch et al., 1995) and the dataset proposed by Boctor (1993). However, these datasets show some shortcomings given the recent evolution in the development of metaheuristic search procedures. We therefore propose a new benchmark dataset MMLIB to overcome the disadvantages of the current datasets. This new dataset can be used by researchers to compare the results of their solution procedures with other procedures. In order to make a fair comparison between all metaheuristic solution procedures on the same computer and for the same stop criteria, we also code each algorithm available in the literature and test their performance on each of the three benchmark datasets.

In Part II of this work, we introduce several practical concepts in order to take steps towards real-life scheduling procedures. One of these practical concepts is preemption, in which is it possible to preempt an activity at any integer time instance and restart it later on at no additional cost. In order to allow activity preemption, the original activity network is converted into a new network, in which each activity is split into subactivities with a unit duration of 1. The introduction of preemption leads to a significant decrease in the average project makespan compared to the non-preempted case.

Another concept deals with the assumption that productivity and efficiency changes can occur during project execution due to the effect of learning, the process of acquiring experience while performing an activity. This concept of activity-specific learning, in which the resources become more efficient the longer they stay on the job, is examined from various angles. The concept is introduced in the discrete time/resource trade-off problem, in which each activity contains a specific work content in terms of working days, instead of a fixed duration and resource requirement. For each activity, a set of execution modes can be specified by using different combinations of durations and resource requirements, as long as the specified work content is met. Computational tests find a significant influence of the introduction of learning effects in project scheduling and reveal the main project drivers that affect the project makespan when learning effects are introduced. Moreover, the margin of error made by ignoring learning during schedule construction is measured. Finally, it is shown that timely incorporating learning during the project progress leads to significant makespan improvements.

In the last chapter of this work, an attempt is made to use the genetic algorithm designed for the MRCPSP to solve a real-life audit scheduling problem, which consists of generating an appropriate audit team schedule for a small Belgian audit firm. Although the algorithm is applied on a simplification of a practical planning problem, the introduction of optimization techniques significantly improves the efficiency of the audit team schedule.

The contribution of this dissertation is threefold. First, three state-of-the-arts metaheuristics are presented to solve the MRCPSP. Our genetic algorithm, artificial immune system and scatter search procedure generated excellent results.

Moreover, the use of problem specific information in the local search process, such as the use of resource scarceness parameters in the scatter search procedure, increased the efficiency of the procedure significantly. Second, several efforts were made to include practical concepts in order to take steps towards real-life scheduling problems. The introduction of preemption and learning led to new interesting insights in project scheduling research. Finally, the new dataset will facilitate and motivate researchers to investigate and develop new ideas and techniques to solve the MRCPSP. Researchers are encouraged to use this dataset to compare the results of their solution procedures with other procedures. The generation of this new dataset makes room for further research on this interesting and challenging research topic.

1 Introduction

As a formal discipline, Operations Research originated as the mathematical scheduling of a massive project logistically supplying Europe with military equipment and goods during WWII (Józefowska and Weglarz, 2006). In the decades after the war, the discipline expanded into a field widely used to solve, analyze and optimize complex problems in business, society and industry. The developed techniques and procedures were and are applied in industries ranging from petrochemistry to airlines, finance, logistics and government, and operations research has become an area of active academic and industrial research (Hillier and Lieberman, 2005).

One of the research fields on which operational research techniques are applied is project management. Project management is the application of knowledge, skills, tools, and techniques to project activities to meet project requirements. Project management is accomplished through the use of the process of initiating, planning, executing, controlling and closing (PMBOK, 2004). A project can be defined as a set of activities with a defined start point and defined end state, which pursues a defined goal and uses a defined set of resources (Slack et al., 2009). The examples of projects are countless. The construction of the Burj Khalifa (the tallest man-made structure ever built), the development of an iPhone application, the building of the pyramids in Egypt, the construction of the wind park on the Thorntonbank, the organization of an electoral campaign or the construction of a house are all examples of projects.

During the past decades, the importance of project management continues to grow rapidly. Many books on project management have been published[1], new project management software is developed, software packages have been expanded with new scheduling capabilities and new techniques for measuring project progress have been developed. The gap between the theory and practice is reduced due to the popularization of project management research. In other words, project management is booming.

Project scheduling has also been an attractive research topic during the past decades. Project scheduling stems from machine scheduling, in which a group of tasks should be assigned to a machine or resource. Different research fields within project scheduling have been explored during recent years and a large set of exact, heuristic and metaheuristic solution procedures have been designed in order to tackle a wide variety of project scheduling problems.

In this work, we study the multi-mode resource-constrained project scheduling problem (MRCPSP), which is a generalization of the resource-constrained project scheduling problem (RCPSP). Due to the resource constraints, this problem is known to be NP-hard (Blazewicz et al., 1983), meaning that optimal solution procedures can only be used for relatively simple problem instances, while (meta)heuristic procedures will be needed for large-sized projects.

For the MRCPSP, many exact, heuristic and metaheuristic solution procedures are proposed in recent years. Each of these procedures is tested on different sets of instances for different stop criteria which makes it difficult to present a fair comparison between the different procedures. The aim of this dissertation is to construct new state-of-the-art metaheuristic solution procedures for the MRCPSP and to make a fair comparison between the different metaheuristic solution procedures available in the literature. Moreover, the current benchmark datasets, which are used to test the efficiency and performance of the solution procedures, show some shortcomings given the recent evolution in the development of metaheuristic search procedures. We therefore propose a new dataset, which aims to overcome the drawbacks of these datasets.

We also introduce several practical concepts in order to take steps towards real-life scheduling procedures. One of these practical concepts is preemption, in which it is possible to preempt an activity at any time and restart it later on at no additional cost. Another concept deals with the assumption that productivity and efficiency changes can occur during project execution due to the effect of learning, which indicates the process of acquiring experience while performing an activity. The introduction of these concepts leads to new interesting insights in project scheduling research. In a case study, we apply one of our metaheuristic approaches on a real-life audit scheduling problem, which consists of generating an appropriate audit team schedule for a small Belgian audit firm.

The remainder of this work is structured as follows. Part I starts with the for-

[1] We refer the interested reader to Vanhoucke (2010), amongst many others.

mulation of the MRCPSP (chapter 2). In chapter 3, three metaheuristic solution procedures for the MRCPSP are proposed, while in chapter 4 a new dataset for the MRCPSP is proposed. In chapter 5, a fair comparison of the different solution procedures is made which gives an indication of the performance of our own procedures and classifies all procedures according to similar stop criteria.

In part II, two extensions on the MRCPSP are presented. In chapter 6, the introduction of preemption is discussed, while in chapter 7, the influence of the introduction of learning effects on the project makespan is investigated. In chapter 8, a model for an audit scheduling problem with sequence-dependent setup times and different audit team efficiencies is proposed and the results for a real-life audit office scheduling problem are presented. In the last chapter, overall conclusions and suggestions for future research are presented.

Publications

Parts of this dissertation have already been presented at international conferences or have been published in international journals.

Publications in international journals

- Van Peteghem, V. and Vanhoucke, M., 2009, ”An Artificial Immune System for the Multi-mode Resource-Constrained Project Scheduling Problem”, Lecture Notes in Computer Science, 5482, 85-96.

- Van Peteghem, V. and Vanhoucke, M., 2010, ”A Genetic Algorithm for the Preemptive and Non-preemptive Multi-mode Resource-Constrained Project Scheduling Problem”, European Journal of Operational Research, 201, 409-418.

Unpublished working papers

- Van Peteghem, V. and Vanhoucke, M., 2009, ”Using Resource Scarceness Characteristics to Solve the Multi-Mode Resource-Constrained Project Scheduling Problem”, Working paper 09/595

- Van Peteghem, V. and Vanhoucke, M., 2010, ”Introducing Learning Effects in Resource-Constrained Project Scheduling”, Working paper 10/633

- Van Peteghem, V. and Vanhoucke, M., 2010, ”An Experimental Investigation of Metaheuristics for the Multi-Mode Resource-Constrained Project Scheduling on New Dataset Instances”, Working paper

Presentations at conferences

- Van Peteghem, V. and Vanhoucke, M., 2007, ”A Genetic Algorithm for the Multi-Mode Resource-constrained Project Scheduling Problem”, Paper presented at the 22^{nd} European Conference on Operational Research – Prague (Czech Republic)

- Van Peteghem, V. and Vanhoucke, M., 2008, ”A Comparison of Various Population-based Meta-heuristics to Solve the MRCPSP”, Paper presentated at the 11^{th} International Workshop on Project Management and Scheduling – Istanbul (Turkey)

- Van Peteghem, V. and Vanhoucke, M., 2009, ”An Artificial Immune System for the Multi-mode Resource-Constrained Project Scheduling Problem”, Paper presented at the 9^{th} European Conference on Evolutionary Computation in Combinatorial Optimization – Tübingen (Germany)

- Van Peteghem, V. and Vanhoucke, M., 2009, ”Introduction of Learning Effects in Resource-constrained Projects”, Paper presented at the 23^{th} European Conference on Operational Research – Bonn (Germany)

- Van Peteghem, V. and Vanhoucke, M., 2010, ”Learning Effects under Different Project Settings”, Paper presented at the 12^{th} International Workshop on Project Management and Scheduling – Tours (France)

- Van Peteghem, V. and Vanhoucke, M., 2010, ”An Experimental Investigation of Meta-heuristics for the Multi-mode Resource-constrained Project Scheduling on new Dataset Instances”, Paper presented at the 24^{th} European Conference on Operational Research – Lisbon (Portugal)

- Van Peteghem, V. and Vanhoucke, M., 2010, ”Audit-staff Scheduling with Alternative Audit Teams and Setup Times”, Paper presented at the 24^{th} European Conference on Operational Research – Lisbon (Portugal)

Part I

Metaheuristics for the MRCPSP

2 The Multi-mode Resource-constrained Project Scheduling Problem

2.1 Introduction

Resource-constrained project scheduling has been a research topic for many decades, resulting in a wide variety of optimization procedures. The main focus on project lead time minimization has led to the development of various exact and (meta)heuristic procedures for scheduling projects with tight resource constraints under a wide variety of assumptions. The basic problem type in project scheduling is the well-known resource-constrained project scheduling problem (RCPSP). This problem type aims at minimizing the total duration or makespan of a project subject to precedence relations between the activities and the limited renewable resource availabilities, and is known to be NP-hard (Blazewicz et al., 1983).

The multi-mode RCPSP (MRCPSP) is a generalized version of the RCPSP, where each activity can be performed in different sets of modes, with a specific activity duration and resource requirements. Three different categories of resources can be distinguished (Slowinski et al., 1994): renewable resources, which are limited per time-unit (e.g. manpower, machines), nonrenewable resources, which are limited for the entire project (e.g. budget) and doubly constrained resources, which are limited both per time-unit and for the total project duration (e.g. cash-flow per time-unit). Since doubly constrained resources can be considered as a combination of renewable and nonrenewable resources, we do not consider them explicitly. The objective of the MRCPSP is to find a mode and a start time for each activity such that the makespan is minimized and the schedule is feasible with respect to the precedence and renewable and nonrenewable resource constraints. As this problem is a generalization of the RCPSP, the MRCPSP is also NP-hard. Moreover, if there is more than one nonrenewable resource, the problem of finding a feasible solution for the MRCPSP is NP-complete (Kolisch and Drexl, 1997). The problem is denoted as $m, 1T|cpm, disc, mu|C_{max}$ using the classification scheme of Herroelen and De Reyck (1999) and is denoted as $MPS|prec|C_{max}$ by Brucker et al. (1999).

In this chapter, an overview is given of the MRCPSP. In section 2.2, a general formulation of the problem is described. In section 2.3, an example project is presented which is used to explain the concepts and terminology which are presented in section 2.4. This chapter concludes with an extended overview of the current literature on the MRCPSP.

2.2 Problem formulation

The MRCPSP can be stated as follows. The project is represented as an activity-on-the-node network $G(N, A)$, where N is the set of activities and A is the set of pairs of activities between which a finish-start precedence relationship with a minimal time lag of 0 exists. A set of activities, numbered from 1 to $|N|$ with a dummy start node 0 and a dummy end node $|N| + 1$, is to be scheduled on a set R^{ρ} of renewable and R^{ν} of nonrenewable resource types. Each activity $i \in N$ is performed in a mode m_i, which is chosen out of a set of $|M_i|$ different execution modes $M_i = \{1, ..., |M_i|\}$. The duration of activity i, when executed in mode m_i, is d_{im_i}. Each mode m_i requires $r^{\rho}_{im_ik}$ renewable resource units ($k \in R^{\rho}$). For each renewable resource $k \in R^{\rho}$, the availability a^{ρ}_k is constant throughout the project horizon. Activity i, executed in mode m_i, will also use $r^{\nu}_{im_il}$ nonrenewable resource units ($l \in R^{\nu}$) of the total available nonrenewable resource a^{ν}_l. A schedule S is defined by a vector of activity start times s_i and a vector denoting its corresponding execution modes m_i. A schedule is said to be feasible if all precedence and renewable and nonrenewable resource constraints are satisfied. The objective of the MRCPSP is to minimize the makespan of the project.

The MRCPSP can be conceptually formulated as follows:

$$\text{Min. } s_{|N|+1} \tag{2.1}$$

s.t.

$$s_i + d_{im_i} \leq s_j \qquad \forall (i,j) \in A \tag{2.2}$$

$$\sum_{i \in S(t)} r^{\rho}_{im_ik} \leq a^{\rho}_k \qquad \forall k \in R^{\rho}, \forall m_i \in M_i \tag{2.3}$$

$$\sum_{i=1}^{|N|} r^{\nu}_{im_il} \leq a^{\nu}_l \qquad \forall l \in R^{\nu}, \forall m_i \in M_i \tag{2.4}$$

$$m_i \in M_i \qquad \forall i \in N \tag{2.5}$$

$$s_0 = 0 \tag{2.6}$$

$$s_i \in \text{int}^+ \qquad \forall i \in N \tag{2.7}$$

where $S(t)$ denotes the set of activities in progress in period $[t-1, t[$, $t \in \{1, ..., s_{|N|+1}\}$. The objective function 2.1 minimizes the total makespan of the project. Constraint set 2.2 takes the finish-start precedence relations with a minimal time lag of 0 into account. Constraints 2.3 and 2.4 take care of the renewable and nonrenewable resource limitations, respectively. Each activity i has to be performed in exactly one mode m_i (constraint 2.5). Constraint 2.6 forces the project to start at time instance 0 and constraint 2.7 ensures that the activity start times assume nonnegative integer values. A schedule which fulfills all the constraints 2.1 to 2.7, is called optimal.

The MRCPSP can be divided into two subproblems: a first subproblem can be referred to as the *Mode Assignment Problem (MAP)*, whose aim is to find a feasible mode assignment. A mode assignment which uses more nonrenewable resources than available is called infeasible, otherwise the mode assignment is called feasible.

The number of requested nonrenewable resource units that exceeds the capacity a^{ν}_l, $l \in R^{\nu}$, is defined as the excess of resource request ERR. The formula of the ERR can be stated as follows:

$$ERR = \sum_{j=1}^{l} (\max(0, \sum_{i=1}^{|N|} (r^{\nu}_{im_ij}) - a^{\nu}_j)) \qquad l \in R^{\nu} \tag{2.8}$$

An ERR=0 means that the solution is feasible. If ERR is larger than 0, the solution is infeasible.

In a second subproblem, the order in which the activities need to be scheduled must be determined. Given the duration and the resource consumptions of the different activities, the aim of the scheduling problem is to minimize the makespan of the project.

2.3 Example

In this section, an example project is presented which will be used throughout the remainder of this chapter in order to explain the concepts and terminology which will be presented in section 2.4. The example project has 8 non-dummy activities, each with 2 modes. For each mode, 1 renewable resource and 1 nonrenewable resource is indicated. The availability for the renewable (nonrenewable) resource is 7 (23). The activity-on-the-node network is shown in figure 2.1. In table 2.1 the duration d_{im_i} and resource requirements ($r^{\rho}_{im_i}$ and $r^{\nu}_{im_i}$) for mode m_i of activity i are shown.

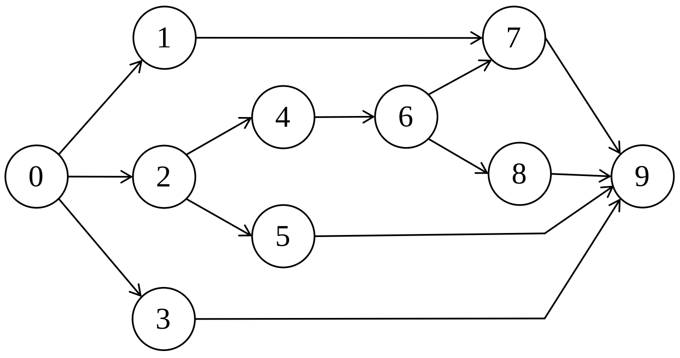


Figure 2.1: Network of the example project

2.4 Definitions

The research conducted in this dissertation builds further on extensive research in project scheduling in the past decades. In order to give a brief introduction to the project scheduling field, different concepts and terminologies which are commonly used in this work are presented in this section. Most of them are explained using the example presented above.

Schedule representation A solution procedure for the (M)RCPSP does not operate directly on a schedule, but on a representation of a schedule that is convenient and effective for the functioning of the algorithm. Kolisch (1999) distinguished 5 different schedule representations in the RCPSP literature, from which the activity list (AL) representation and the random key (RK) representation are the most widespread.

Activity list In the AL representation, the position of an activity in the AL determines the relative priority of that activity versus the other activ-

act i	mode m_i	d_{im_i}	$r^{\rho}_{im_i}$	$r^{\nu}_{im_i}$
0	1	0	0	0
1	1	4	3	3
	2	5	2	4
2	1	1	3	4
	2	2	2	3
3	1	1	2	3
	2	2	1	1
4	1	2	5	4
	2	3	4	3
5	1	2	4	6
	2	5	3	2
6	1	1	1	4
	2	3	1	3
7	1	1	3	3
	2	3	2	2
8	1	2	3	4
	2	2	3	3
9	1	0	0	0
Available			7	23

Table 2.1: Information of the example project

ities. In table 2.2, an example of an activity list is given. In order to avoid infeasible solutions, the activity list is always precedence feasible, which means that the precedence relations between the different activities are met.

Random Key In the RK representation, the sequence in which the activities are scheduled is based on the priority value attributed to each activity. It is assumed in this work that a low RK value corresponds to a high priority. In table 2.3, three random key representations are given, which will all result in the same project schedule.

place	1	2	3	4	5	6	7	8
AL	1	2	3	4	6	7	5	8

Table 2.2: Activity list

Mode representation Next to the schedule representation, the mode representation determines the execution mode of each activity. Once a mode is assigned to an activity, the duration and the resource consumption of each

activity	1	2	3	4	5	6	7	8
RK_1	12	16	19	25	21	18	12	28
RK_2	1	2	3	5	4	6	7	8
RK_3	0.21	0.25	0.98	1.15	1.02	2.21	0.24	0.25

Table 2.3: Random key

resource type can be determined. Two mode representations can be distinguished: the mode list and the mode vector.

Mode list In the mode list representation, the list represents the execution modes of the activities in ascending order, i.e. the first number in the list indicates the mode in which the first activity will be executed, the second number the execution mode of the second activity, etc. In table 2.4, an example of a mode list is given. Activity 7, for example, is executed in mode 1.

Mode vector In the mode vector representation, the mode indicated in the i^{th} position of the mode vector represents the execution mode of the activity placed in the i^{th} position of the activity list. A mode vector is always represented in combination with an activity list. In table 2.5, an example of a mode vector is given, together with the activity list as represented in table 2.2. As can be seen, the same modes are chosen for each activity as in the mode list example.

activity	1	2	3	4	5	6	7	8
mode list	2	1	2	1	2	1	1	2

Table 2.4: Mode list

place	1	2	3	4	5	6	7	8
AL	1	2	3	4	6	7	5	8
MV	2	1	2	1	1	1	2	2

Table 2.5: Mode vector

Mode reduction Before the mode lists are generated, the mode reduction procedure of Sprecher et al. (1997) can be applied. This procedure excludes those modes which are inefficient or non-executable and those resources which are redundant.

- A mode is called *inefficient* if there is another mode of the same activity with the same or smaller duration and no more requirements for all resources.
- A mode is called *non-executable* if its execution would violate the renewable or nonrenewable resource constraints in any schedule.
- A nonrenewable resource is called *redundant* if the sum of the maximal requests for that nonrenewable resource does not exceed its availability.

Excluding these modes or nonrenewable resources does not affect the set of feasible or optimal schedules.

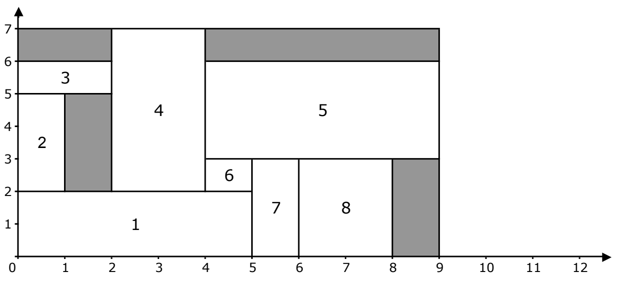
Consider the example project instance given in section 2.3. Mode 1 of activity 8 can be called inefficient because both its duration and its resource requirements are equal or larger than those of mode 2. Also mode 1 of activity 5 can be excluded, because this mode is non-executable with respect to the nonrenewable resource. If it were executed, the project would require at least 24 nonrenewable resource units, while only 23 are available. The mode can therefore be deleted.

Schedule generation scheme A schedule generation scheme (SGS) translates the schedule representation into a schedule. Two different types of SGSs exist in the literature: the serial SGS (Kelley Jr., 1963) and the parallel SGS (Bedworth and Bailey, 1982).

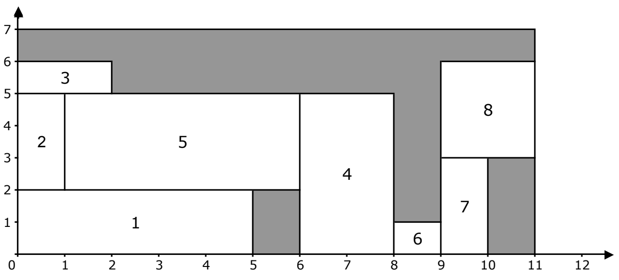
Serial SGS The serial scheduling scheme sequentially adds activities to the schedule one-at-a-time. In each iteration, the next activity in the priority list (activity list or random key) is chosen and that activity is assigned to the schedule as soon as possible within the precedence and resource constraints.

Parallel SGS In contrast to the serial SGS, the parallel scheduling scheme iterates over the different schedule times t_s in which activities can be added to the schedule. These schedule times correspond to the completion times of already scheduled activities. At each time t_s, the unscheduled activities whose predecessors have been completed, are considered in the order of the priority list and are scheduled on the condition that no resource conflict originates at that time instant.

Figure 2.2(a) depicts a schedule which is based on the activity list as proposed in table 2.2 and the mode list as proposed in table 2.4. The schedule has a makespan of 9 days and is generated by using a serial schedule generation scheme. In figure 2.2(b), the schedule based on the same activity and mode list is shown using a parallel SGS. This schedule results in a makespan of 11 days. However, both schedules are infeasible with respect to the nonrenewable resource since 25 nonrenewable resource units are used, while only 23 nonrenewable resources are available ($ERR = 2$).



(a) Infeasible schedule (serial SGS)



(b) Infeasible schedule (parallel SGS)

Figure 2.2: Schedules of the example project

In the remainder of this example, we use the serial SGS to generate schedules. Figure 2.3(a) shows a schedule with a makespan of 13 days. The schedule is based on an activity list (1,2,3,5,4,6,7,8) and a mode list (2,2,2,2,2,2,1,2). This schedule is feasible because the required nonrenewable resource units do not exceed the availability (ERR = 0) and because all precedence relations are met.

Forward-backward scheduling technique This technique, proposed by Li and Willis (1992), transforms left-justified schedules (where all activities are scheduled as soon as possible) into right-justified schedules (where all activities are scheduled as late as possible) and vice versa. During the backward (forward) scheduling stage, the makespan of the schedule is tried to be reduced by shifting the activities, in a sequence determined by the finish (start) times, as much as possible to the right (left), without affecting the project completion time. During the forward-backward procedure only improvements can occur.

On the schedule presented in figure 2.3(a), the backward scheduling technique is applied. Activities 8 cannot be scheduled later, while activity 7 can be shifted (1 time unit) to start at time instant 12. Activities 6 and 4 cannot be scheduled later. Activity 5 can be right-shifted to start at time 7. Since the right shift of activity 5 has made some additional resources available, activity 1 can be shifted 2 time units to start at time instant 2. Activity 3 can also be shifted to its latest start time 11. In this way, we obtain the schedule of figure 2.3(b) with a makespan of 11 units. Further improvements of the schedule are possible by shifting activities as much as possible to the left (forward scheduling). However, the total makespan of the schedule remains 11, as can be seen in figure 2.3(c).

Crossover During a crossover operation, information of two solution vectors is combined in order to generate a new solution vector. The one- and two-point crossover are the most used crossovers.

One-point crossover In a one-point crossover, a single crossover point is selected and all data beyond that point in either string are swapped between the two parents.

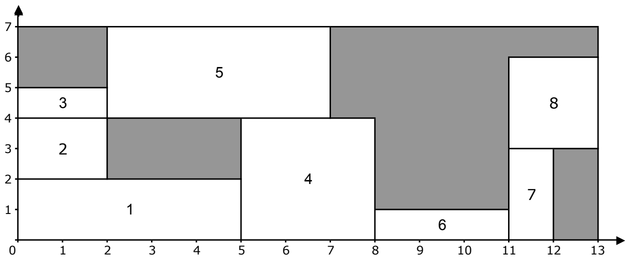
Two-point crossover In a two-point crossover operation, two crossover points are selected on the parent strings and everything between these two crossover points is swapped.

Mutation The mutation operator is applied in order to introduce lost genetic material into the population and creates variation in the different individuals.

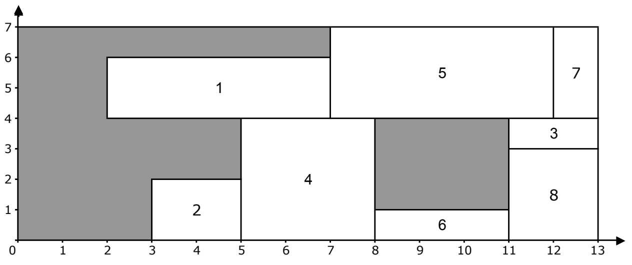
In table 2.6, an example of a one-point crossover is presented. Two activity lists (AL_1 and AL_2) are used to create a new activity list AL_{cross}. The crossover point is chosen randomly and is equal to 3, which means that the first 3 activities are chosen out of activity list AL_1 and the last 5 are selected from activity list AL_2, i.e. the activities which are not selected yet from AL_1 are scheduled following the sequence in AL_2. The new activity list AL_{cross} is presented below. On this new activity list, a mutation operation is applied. The activities 3 and 5 from the activity list AL_{cross} are swapped. This results in the activity list AL_{mut}.

place	1	2	3	4	5	6	7	8
AL_1	**1**	**2**	**4**	5	6	3	7	8
AL_2	2	1	**3**	4	**6**	**5**	**8**	**7**
AL_{cross}	1	2	4	3	6	5	8	7
AL_{mut}	1	2	4	5	6	3	8	7

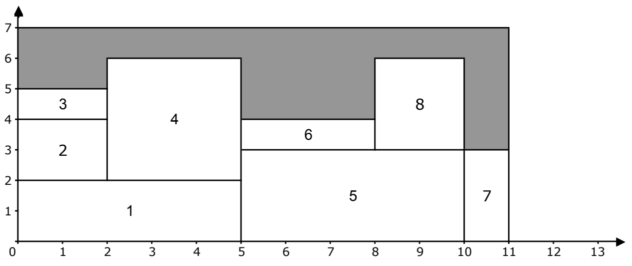
Table 2.6: Crossover and mutation operator



(a) Feasible (left-justified) schedule MRCPSP



(b) Backward scheduling of schedule (a)



(c) Forward scheduling of schedule (b)

Figure 2.3: Schedules of the example project

Instance parameters Several parameters are defined in order to measure the complexity of the project and the scarceness of the resources. Although different parameters are proposed in recent years, we define the ones which will be used in this work.

Order strength The network complexity is described by the order strength (Mastor, 1970), which is defined as the number of precedence relations (including the transitive ones but not including the arcs connecting the dummy start or end activity) divided by the theoretical maximum number of precedence relations ($|N|(|N|-1)/2$). The resource strength OS varies between 0 and 1. An OS close to 0 indicates a parallel network, while an OS close to 1 implies a serial network. The order strength of our example project is equal to 0.39.

Resource strength In the literature, two of the most used parameters to calculate the scarceness of the resources for single-mode projects are the *Resource Strength* (RS), introduced by Cooper (1976) and later on redefined by Alvarez-Valdes and Tamarit (1989) and Kolisch et al. (1995), and the *Resource Constrainedness* (RC), proposed by Patterson (1976). Since no formula is known for the resource constrainedness as a resource parameter for multi-mode resource-constrained projects, we will use in this work the resource strength as a parameter to calculate the scarceness of the renewable and nonrenewable resources. Kolisch et al. (1995) and Demeulemeester et al. (2003) defined the resource strength for multi-mode projects as follows:

$$RS_k = \frac{a_k - r_k^{min}}{r_k^{max} - r_k^{min}} \tag{2.9}$$

where a_k denotes the total availability of renewable resource type k, r_k^{min} is formulated as $\max_{i=1,\ldots,|N|;m_i=1,\ldots,|M_i|} r_{ikm_i}^{\rho}$ and r_k^{max} denotes the peak demand of renewable resource type k in the precedence preserving the earliest start schedule, where each activity has a duration which corresponds to a maximum allocation of resources (Demeulemeester et al., 2003). The resource strength RS varies between 0 and 1. A RS close to 0 indicates that the scarceness of the resource is high, while a RS close to 1 implies that the resource is hardly restrictive. In the example project, the renewable resource strength RS^{ρ} is equal to 0.29. Kolisch et al. (1995) defined r_k^{min} as $\max_{i=1,\ldots,|N|}\left\{\min_{m_i=1,\ldots,|M_i|} r_{ikm_i}^{\rho}\right\}$. However, for low values of RS_k, the use of this definition will lead to different non-executable modes, which means that its execution would violate the renewable (or nonrenewable) resource constraints in any schedule (Sprecher, 2000).

For the nonrenewable resources the minimum and maximum consumption can be obtained by cumulating the consumptions obtained when performing each activity in the mode having minimum and maximum consumptions. In the example project, the value of the nonrenewable resource strength RS^{ν} is equal to 0.25.

Resource factor The resource factor (RF) indicates the percentage of resources that are required per activity. The renewable resource factor RF^{ρ} of resource k can be calculated as follows:

$$RF_k^\rho = \frac{1}{|N|}\sum_{i=1}^{|N|}\frac{1}{|M_i|}\sum_{m_i=1}^{|M_i|}\begin{cases} 1 & \text{if } r_{im_ik}^\rho > 0 \\ 0 & \text{otherwise}\end{cases} \tag{2.10}$$

The nonrenewable resource factor RF^ν can be calculated similarly. In the example project, both the renewable and nonrenewable resource factor is equal to 1.

In this section, different different concepts and terminologies which are commonly used in the project scheduling field are presented. In the following section, an overview is given of the solutions procedures for the MRCPSP currently available in the literature.

2.5 Literature overview

Several exact and heuristic approaches to solve the MRCPSP have been proposed in recent years. In section 2.5.1, an overview is given of the exact solution procedures. In section 2.5.2, we discuss the heuristic solution procedures and in section 2.5.3, we describe the metaheuristic solution procedures. In this last section, each solution procedure is also described in detail, since the computational results of these algorithms will be compared with the metaheuristic solution procedures proposed in this work.

Although the solution procedures that tackle the MRCPSP/max, in which minimal and maximal time lags are incorporated, show many similarities with the procedures proposed for the MRCPSP, we refer in this section only to the solution procedures for the classical MRCPSP. The interested reader is referred to the paper of Barrios et al. (2009) for an overview of the available metaheuristics for the MRCPSP/max.

2.5.1 Exact solution procedures

The first solution method for the multi-mode problem can be found in Slowinski (1980), who presented a one-stage and two-stage linear programming approach. Talbot (1982) and Patterson et al. (1989) presented an enumeration scheme-based procedure. Speranza and Vercellis (1993) proposed a depth-first branch-and-bound algorithm, but Hartmann and Sprecher (1996) have shown that this algorithm may be unable to find the optimal solution for instances with two or more renewable resources. More recently, Sprecher et al. (1997), Hartmann and Drexl (1998) and Sprecher and Drexl (1998) presented branch-and-bound algorithms, while Zhu et al. (2006) proposed a branch-and-cut algorithm. However, none of these procedures can be used for solving large-sized realistic projects, since they are unable to find an optimal solution in a reasonable computation time. Therefore, different single-pass heuristic and metaheuristic procedures are presented.

2.5.2 Heuristic solution procedures

Talbot (1982) and Sprecher and Drexl (1998) proposed to impose a time limit on their exact branch-and-bound procedure. Boctor (1993) tested 21 heuristic scheduling rules and suggested a combination of 5 heuristics which have a high probability of giving the best solution. Drexl and Grünewald (1993) proposed a biased random sampling approach, while Özdamar and Ulusoy (1994) proposed a local constraint based analysis approach. Boctor (1996) presented a heuristic algorithm based on the Critical Path Method computation, Kolisch and Drexl (1997) suggested a local search method with a single neighborhood search, Knotts et al. (2000) evaluated different agent-based algorithms and Lova et al. (2006) designed several multi-pass heuristics based on priority rules for solving the MRCPSP.

2.5.3 Metaheuristic solution procedures

This section gives an overview of the current available metaheuristics from the literature. In section 2.5.3.1 an overview of the different classification criteria, as mentioned in Kolisch and Hartmann (1999), is given and the available algorithms are classified according to these criteria. In section 2.5.3.2 an extensive and detailed overview of all the metaheuristics is presented.

2.5.3.1 Classification criteria

In order to make a classification of the available metaheuristics, the procedures are sorted based on three classification criteria as proposed by Kolisch and Hartmann (1999): the metaheuristic strategy, the schedule representation, the mode representation and the schedule generation scheme. In what follows we briefly examine each of these criteria.

Metaheuristic strategy Several metaheuristic strategies to solve a scheduling problem are available. For an overview of these metaheuristic strategies we refer to Glover and Kochenberger (2003). For the MRCPSP the following six strategies were used: genetic algorithm (GA), scatter search (SS), simulated annealing (SA), particle swarm (PS), tabu search (TS) and artificial immune system (AIS).

Schedule representation Kolisch (1999) distinguished 5 different schedule representations in the RCPSP literature, from which the activity list (AL) representation and the random key (RK) representation are the most widespread.

Mode representation Two mode representations can be distinguished in the literature: the mode vector (MV) and the mode list (ML).

Schedule generation scheme A schedule generation scheme (SGS) translates the schedule representation into a schedule. Two different types of SGSs exist in the literature: the serial SGS (Kelley, 1963) and the parallel SGS (Bedworth and Bailey, 1982). Kolisch (1996b) has shown that it is sometimes impossible to reach an optimal solution with the parallel SGS. Nevertheless, both schemes are used in the solution procedures currently available in the literature.

In table 2.7 an overview of the different metaheuristic algorithms is given. For each solution procedure, the name of the author(s) and the abbreviation used in this work to refer to the procedure is given. The indication R or RNR in the third column indicates if the procedure is applicable on datasets with only renewable resources (R) or with both renewable and nonrenewable (RNR) resources. The information in the fourth, fifth, sixth and seventh column indicates the metaheuristic strategy, the schedule representation, the mode representation and the schedule generation scheme used to solve the problem instances, respectively.

2.5.3.2 Metaheuristic solution procedures

In this section, an overview is given of the different procedures available in the literature according to their metaheuristic strategy: genetic algorithm, simulated annealing, tabu search, scatter search and particle swarm.

Genetic algorithms Introduced by Holland (1975), genetic algorithms (GAs) use techniques and procedures inspired by evolutionary biology to solve complex optimization problems. Several selection mechanisms, such as natural selection, crossover and mutation, are applied in order to recombine existing solutions so that new ones are obtained and an optimal solution is found.

Mori and Tseng (1997) were the first to develop a genetic algorithm for the MRCPSP/R. The algorithm is based on the priority list representation, where the chromosome provides information about the scheduling order and the execution mode for each activity. The scheduling order is the priority of the activity in the schedule and lies between its forward and backward scheduling order (see Tavares, 1990). The crossover operator is a one-point crossover, which randomly chooses an activity for which the start time is lower, and is applied on both the scheduling order and mode list, while the mutation operator randomly adapts the mode list of a randomly chosen schedule. The population of a new generation is generated by duplicating the best offspring schedules, by producing new schedules using the crossover and mutation operator and by generating new random schedules.

Table 2.7: Classification metaheuristics

Author	Abbr	R/RNR	Strategy	Schedule repr	Mode repr	SGS
Slowinski et al., 1994	SLOW	RNR	SA	AL	ML	P
Boctor, 1996a	BOCT	R	SA	AL	ML	S
Mori and Tseng, 1997	MORI	R	GA	RK	ML	S
Özdamar, 1999	OZDA	RNR	GA	RK	ML	P
Nonobe and Ibaraki, 2001	NONO	RNR	TS	AL	ML	P
Jozefowska et al., 2001	JOZE	RNR	SA	AL	ML	S
Hartmann, 2001	HART	RNR	GA	AL	MV	S
Bouleimen and Lecocq, 2003	BOUL	RNR	SA	AL	MV	S
Alcaraz et al., 2003	ALCA	RNR	GA	AL	MV	S
Zhang et al., 2006	ZHAN	RNR	PS	RK	ML	S
Jarboui et al., 2008	JARB	RNR	PS	RK	ML	S
Ranjbar et al., 2008	RANJ	RNR	SS	AL	MV	S
Lova et al., 2009	LOVA	RNR	GA	AL	MV	P/S
Tseng and Chen, 2009	TSEN	RNR	GA	AL	MV	S

Özdamar (1999) presented a hybrid genetic algorithm which makes use of the priority list representation and a parallel SGS that performs exactly two iterations, one forward and one backward on each chromosome. A chromosome is represented by two lists: a mode assignment list and a priority assignment list, which indicates the priority rule used to select the candidate activity in the n^{th} position. For each chromosome a crossover probability is calculated which is dependent on both the chromosome's makespan and the population's makespan and which determines the probability of the chromosome to be recombined in the next generation. A similar calculation is made to determine the probability to apply the mutation operator. In order to generate the population of a new generation, each chromosome is reproduced a number of times proportional to its objective function value.

Hartmann (2001) developed a genetic algorithm based on the activity list representation and a serial SGS, which only generates forward schedules. A two-point crossover operator and a mutation operator are applied with a certain probability in order to create new genes. The ranking method is used as selection operator. The algorithm is extended with two local search procedures to improve the schedules. The single-pass improvement procedure checks for every activity whether a multi-mode left shift can be performed. A multi-mode left shift of an activity j is an operation on a given schedule which reduces the finish time of activity j without changing the modes or finish times of the other activities and without violating the precedence and resource constraints. The multi-pass improvement procedure repeats the single-pass procedure until no further improvements can be detected. However, computational results have shown that the single-pass improvement procedure performs the best.

Alcaraz et al. (2003) developed a genetic algorithm based on the activity list representation and the serial SGS. An additional gene decides whether a forward or backward scheduling is employed when computing a schedule from an activity and mode list. The two-point forward-backward crossover operator as designed by Alcaraz and Maroto (2001) is extended to the multi-mode version. A two-phase mutation operator is applied, which firstly modifies the activity list and secondly alters the mode assignment.

Lova et al. (2009) proposed a hybrid genetic algorithm, with an activity list representation and a serial and parallel SGS, which generates forward and backward schedules. Two extra genes decide whether a forward or backward scheduling is employed and whether the serial or parallel SGS is executed. The two-point crossover operator and mutation are applied and the two-tournament selection is applied to reproduce the new population. The authors also introduced a multi-mode forward-backward improvement method, which is an extension of the forward-backward improvement method described by Tormos and Lova (2001) and which can change the execution mode if a better position for an activity can be found.

Tseng and Chen (2009) presented a two-phased genetic local search algorithm, with an activity list representation and a serial SGS, which generates forward and backward schedules. During the first phase, a set of elite solutions is searched by the genetic algorithm, using a modified two-point crossover operator and a mutation operator, based on the critical path activities in the schedule. This elite set is utilized to construct the initial population of the second phase. A mutation operator and a forward-backward local search is applied on this elite set, in order to search more thoroughly in the promising areas of the solution space.

Simulated annealing The simulated annealing method is based on the physical annealing process and has been introduced for the first time by Metropolis et al. (1953). In this method, all improvements are accepted, while inferior solutions are rejected or accepted with a certain probability, which decreases with the value of the difference in costs of the current and neighbor solution. The method has been used several times to solve the MRCPSP.

Boctor (1996) was the first to present a simulated annealing procedure and presented a procedure with an activity list representation and a serial SGS for the RCPSP and MRCPSP/R. The procedure started with an initial solution, generated by the minimum slack/shortest feasible mode heuristic as proposed in Boctor (1993). To generate neighbor solutions, a random chosen activity is moved to another position in the precedence feasible activity list. To determine the execution mode, the mode resulting in the earliest finish time taking into account the precedence and renewable resource constraints is chosen. Several heating, reheating and cooling phases are proposed.

Slowinski et al. (1994) proposed a simulated annealing procedure with an activity list representation and a parallel SGS. The initial starting solution is the best solution among a set of parallel priority heuristics. A neighborhood solution is accepted with a probability of exp(-ρ/T), where T is the control parameter, determined by the acceptable deterioration rate, and ρ the actual deterioration of the solution vector.

Bouleimen and Lecocq (2003) proposed a simulated annealing procedure with an activity list representation and a serial SGS. The authors used two separate exploration techniques in a two-stage procedure. In the first stage only a mode neighborhood exploration is performed, while in the second stage an activity neighborhood exploration is performed in order to further improve the solution. However, the second stage is only applied if a smaller makespan is found in the first stage. The initial value of the procedure is generated randomly. In the first exploration phase, no probabilistic acceptance criteria were used and only neighbors with a smaller makespan were accepted. In the second phase, fixed control parameters were used in order to fully explore the search space.

Józefowska et al. (2001) proposed a simulated annealing procedure with an activity list representation and a serial SGS, which makes use of the adaptive cooling scheme of Aarts and Korst (1989). In this work, the value of the control parameter is variable since the value depends on the search path. This involves the absence of a reheating phase, as used in the papers of Boctor (1996) and Bouleimen and Lecocq (2003). The initial solution can be obtained by setting all activities on the activity list in an ascending order that follows from the ordering of nodes in the precedence relation graph, and by executing all jobs in their first modes. The generation of a neighborhood is generated by using one of the following operators: a neighborhood shift which operates only on the list of activities, a mode change which operates only on the mode list or a combined move.

Tabu search Tabu Search (TS) is a local search method, designed to drive the search away from local optima by accepting non-improving solutions. An intelligent use of memory is employed to help in exploiting the characteristics of previous solution runs. The structure of the method is extremely malleable and hence it is often used to guide other constructive heuristics in order to avoid being trapped into a poor local optimum.

Nonobe and Ibaraki (2002) presented a tabu search procedure for the RCPSP and the MRCPSP with an activity list representation and a parallel SGS. In the initial part of the tabu search procedure, solutions are randomly generated in order to find a feasible solution. Afterwards, this solution is repeatedly replaced by its best non-tabu neighbor until the stop criterion is achieved. Three types of neighborhood moves are proposed: a mode shift $change\_mod(i, m'_i)$, which changes the mode of activity i to mode m'_i; an activity shift $shift\_aft(i, j)$, which shifts activity i immediately after activity j and an activity shift $shift\_bef(i, j)$, which modifies the position of activity j to the position before i. The tabu list prohibits all moves executed in the τ recent iterations, where τ is a program parameter called tabu tenure.

Scatter search Scatter search is a population-based metaheuristic, proposed by Glover et al. (2000), in which solutions are intelligently combined to yield better solutions. The scatter search method makes use of deterministic procedures that can include problem specific knowledge (Pinol and Beasley, 2006) and can therefore be implemented in a variety of ways and degrees of sophistication. For an overview of the basic and advanced features of the scatter-search, we refer to Glover et al. (2000) and Marti et al. (2006).

Ranjbar et al. (2009) presented a scatter search algorithm with an activity list representation and a serial SGS. In order to start with a diverse initial population of solutions, the biased random sampling using frequency memory is employed. The solutions are combined using the path relinking method (Glover et al., 2000).

This approach generates new solutions by exploring trajectories connecting high-quality solutions. Finally, a local search procedure, which changes with a certain probability the mode m_i of an activity i to $m_i - 1$ and $m_i + 1$ while the other activities remained unchanged, is applied to each generated schedule.

Particle Swarm Particle Swarm optimization (PS) was introduced by Kennedy and Eberhart (1995) and simulates the swarming behavior of animals to reach the promising areas. Just like a population-based metaheuristic, PS conducts a search using a population (called swarm) of individuals (called particles) that is updated from iteration to iteration, using formulas for each particle's position and velocity. Two authors have used the principles of particle swarm to solve the MRCPSP.

The PS algorithm of Zhang et al. (2006) makes use of the priority list representation and a serial SGS. The initial population and the initial velocities are generated randomly. The formulas of Kennedy and Eberhart (1995) are used to update the population. In order to adjust infeasible particle solutions, a solution procedure is proposed to change the infeasibility of the current solution based on the activity's priority value. However, one can prove that endless loops can occur due to the randomness of the mode selection part.

Jarboui et al. (2008) proposed a combinatorial PS algorithm with a priority list representation and a serial SGS. The generation of the initial mode lists is generated according to a certain probability (per mode) which increases for decreasing renewable resource consumption of that mode. Once a mode list is generated, a local search optimization procedure is applied on the list to optimize the sequence in which the activities should be scheduled. The activities are scheduled according to a probability which is determined by its number of successors. Feasible swaps are then performed in order to obtain improvements in the best solution.

2.6 Conclusions

In this chapter, an introduction is given to the MRCPSP. We described the general formulation of the problem and presented an example project which is used to explain the most important concepts and terminology used in project scheduling literature. This chapter concluded with an overview of the available literature on the MRCPSP and an introduction to each available metaheuristic solution procedure.

3 Metaheuristic Solution Procedures for the MRCPSP

3.1 Introduction

During the past decades, different metaheuristic solution procedures were used to solve a wide set of combinatorial optimization problems. Each of these metaheuristics is composed of different techniques, which are not dedicated to the solution of a particular problem, but are designed with the aim to be flexible enough to handle a wide range of combinatorial problems. A metaheuristic can therefore be considered as a conceptual framework which can be adapted with a few modifications to fit a specific optimization problem. Metaheuristic strategies guide and modify the operations of subordinate heuristics and explore the search space in order to find (near-)optimal solutions (Osman and Laporte, 1996).

The advantage of metaheuristic algorithms is the interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust search of a solution space (Glover and Kochenberger, 2003). Local improvement methods attempt to explore intensively the promising regions in the neighborhood of the current solution, while the search strategies of the different metaheuristic philosophies are designed to exploit the entire solution space. This balance between intensification and diversification is very important for finding the near-optimal solution efficiently. On the one hand, the search process should quickly identify these regions in the search space where high quality solutions can be found, while on the other hand, the algorithm should not waste too much time in regions that are either already explored or do not provide high quality solutions (Blum and Roli, 2003). Since metaheuristics are not problem-specific and are composed out of a set of basic concepts, it is often necessary to include problem specific adaptions to these methods in order to obtain an effective metaheuristic solution procedure.

There are different metaheuristic solution procedures available in the literature. In the previous chapter, the genetic algorithm, the scatter search, the simulated annealing, the particle swarm and tabu search procedures were already presented, but many others are available (Glover and Kochenberger, 2003). According to Blum and Roli (2003), metaheuristic philosophies can be classified and described in different ways: *nature-inspired* versus *non-nature inspired*, *one* versus *various neighborhood structure*, *memory usage* versus *memory-less methods* and *single-point* versus *population-based*. This last classification is based on the characteristic of the number of solutions used at the same time. Algorithms working on one single solution are called *trajectory methods* and encompass metaheuristics like tabu search, iterated local search and variable neighborhood search. *Population-based metaheuristics*, on the contrary, perform search processes that describe the evolution of a set of points in the search space.

Osman (1995) makes another division and classifies the family of metaheuristics into three categories. The first is that of *construction-based metaheuristics*, which include greedy random adaptive search methods, guided construction methods and ant colony systems. These metaheuristics tackle an optimization problem by exploring the search space using a so-called search tree. Each path from the root node of the search tree to one of the leaves corresponds to the process of constructing a candidate solution. The metaheuristics gradually build a solution by sampling the search space in every iteration. The second category is that of *local-search-based metaheuristics*, which include simulated annealing, noisy methods, guided local search methods, iterated local search, neural networks, tabu search, threshold accepting and variable neighborhood search. These methods include intelligent extensions of local search algorithms in order to escape from local minima to proceed with the exploration of the search space. The third category is that of *population-based metaheuristics*, which include evolutionary methods (genetic algorithms, memetic algorithms, artificial immune systems, ...), path-relinking and scatter search. Since this type of metaheuristics deals with a population of solutions, population-based algorithms provide an intrinsic way for the exploration of the search space.

In this chapter, three different population-based metaheuristic solution procedures for the MRCPSP are proposed, i.e. a genetic algorithm, an artificial immune system and a scatter search. Before the metaheuristics are described in detail, each of the solution procedures is classified based on different classification criteria in section 3.2. Next, in section 3.3, a bi-population genetic algorithm is described, which makes use of two separate populations and extends the serial schedule generation scheme by introducing a mode improvement procedure. In section 3.4, an artificial immune system algorithm is presented. This algorithm makes use of mechanisms which are inspired by the vertebrate immune system, such as hypermutation and proliferation. The initial population set is generated with a controlled mode assignment procedure, based on experimental results which reveal a link between predefined mode list characteristics and the project makespan. Finally, in section 3.5, a scatter search algorithm is proposed, which is executed with different improvement methods, each tailored to the specific characteristics of different renewable and nonrenewable resource scarceness values. These resource parameters have been introduced in project scheduling literature to measure the scarceness of resources of a project instance and are incorporated in the search process of the scatter search procedure.

Each section describes the different metaheuristics in detail and provides some computational results for the configuration of the solution procedure. In chapter 5, a comparison is made between the different metaheuristic solution procedures. Computational tests are therefore performed on three datasets: the PSPLIB dataset (proposed by Kolisch et al., 1995), the dataset set by Boctor (1993) (hereafter called the Boctor dataset) and the MMLIB dataset, which is the new benchmark dataset presented in chapter 4.

3.2 Classification

In order to give an extensive overview of the different components which will be used in the three metaheuristics, each solution procedure is classified according to nine metaheuristic classification criteria. Some of these criteria are inherent to the metaheuristic solution procedure (e.g. the number of populations in a scatter search procedure), some are straightforward (e.g. the metaheuristic strategy), while the performance of some are tested during the computational experiments (e.g. the penalty function). In section 3.2.1, an overview of the classification criteria is given, while in section 3.2.2, the classification for the three metaheuristics is made and discussed.

3.2.1 Classification criteria

The classification for each of the metaheuristic procedures is based on the following nine criteria. For the first three classification criteria, we also refer to section 2.5.3.1.

Metaheuristic strategy Three metaheuristic procedures are proposed in this chapter: a genetic algorithm, an artificial immune system procedure and a scatter search procedure. A genetic algorithm uses several mechanisms, such as natural selection, crossover and mutation in order to recombine existing solutions so that new ones are obtained. The scatter search algorithm contrasts with the genetic algorithm by focusing not only on the quality of the solution, but also on the diversity of a solution. The use of diversification and improvement methods leads to a more efficient exploration of the solution space. The artificial immune system finally makes use of mechanisms, such as the clonal selection process, hypermutation and receptor editing, which are inspired by the vertebrate immune system.

Schedule representation Two schedule representations are used: the activity list (AL) representation, where the position of an activity in the AL is determined by the relative priority of that activity versus the other activities, and the random key (RK) representation, where the sequence in which the activities are scheduled is based on the priority value attributed to each activity.

Schedule generation scheme A schedule representation can be translated into a schedule by a serial SGS or by a parallel SGS. Since Kolisch (1996b) has shown that it is sometimes impossible to reach an optimal solution with the parallel SGS, we use in our metaheuristic solution procedures only the serial SGS.

Penalty function As infeasible solutions can be included in the population, infeasible schedules must be penalized. Therefore, several penalty functions are available in the literature: the penalty function of Hartmann (2001), the penalty function of Alcaraz et al. (2003), the penalty function of Jarboui et al. (2008) and the penalty function of Lova et al. (2009).

Initial population Since the three proposed solution procedures are population-based metaheuristics, the algorithm needs an initial population on which the algorithmic techniques can be applied. This initial population can be generated randomly or can be generated by the controlled mode assignment procedure, which will be proposed in section 3.4.

Local improvement methods Local improvement methods attempt to intensively explore the promising regions in the neighborhood of the current solution. Józefowska et al. (2001), Bouleimen and Lecocq (2003), Kolisch and Drexl (1997) and Hartmann (2001) already proposed several local search procedures for the MRCPSP, which will be referred to as 'Local search literature'. In section 3.3, a mode improvement method will be described and a combined local improvement method, which is composed out of three improvement methods, will be presented in section 3.5.

Number of populations Population-based algorithm traditionally use 1 population on which different procedures are applied. However, in this work, we will propose algorithms which make use of 2 populations.

Crossover Although different crossover operators are proposed in the literature, tests revealed that the one-point and two-point crossover obtained the best results in our algorithms.

Mutation In one procedure no mutation operator is applied. Two classification items are therefore considered: mutation or no mutation.

3.2.2 Classification of the proposed metaheuristics

In table 3.1, an overview of the different classification criteria is given. For each of the developed metaheuristic solution procedures, it is indicated if and how this classification criterion is used in the proposed algorithm ('x' means the component is applied and/or tested in this solution procedure).

During the computational experiments of the genetic algorithm (section 3.3), the performance of the different penalty functions, the number of populations and the local improvement methods from the literature are tested. During the computational experiments of the AIS (section 3.4), the initial population generation methods (controlled versus random) are tested. The scatter search algorithm makes use of a combination of the best performing local improvement methods from the literature, the mode improvement method and the combined improvement method. This is tested in section 3.5.

In the remainder of this chapter, a detailed description of each metaheuristic solution procedure is given.

	GA	AIS	SS
Metaheuristic strategy			
Genetic algorithm	x	-	-
Artificial Immune System	-	x	-
Scatter Search	-	-	x
Schedule representation			
Random Key	x	-	x
Activity list	-	x	-
Schedule generation scheme			
Serial	x	x	x
Parallel	-	-	-
Penalty function			
Hartmann (2001)	x	-	-
Alcaraz et al. (2003)	x	x	x
Jarboui et al. (2008)	x	-	-
Lova et al. (2009)	x	-	-
Initial population			
Randomly	x	x	-
Controlled mode assignment procedure	-	x	x
Local improvement methods			
Local search literature	-	-	x
Mode improvement method	x	-	x
Combined improvement method	-	-	x
Number of populations			
One population	x	x	-
Two populations	x	-	x
Crossover operator			
One-point crossover	x	-	-
Two-point crossover	-	-	x
Mutation operator			
Mutation	x	x	-
No mutation	-	-	x

Table 3.1: Classification of the metaheuristics

3.3 Genetic algorithm

3.3.1 Introduction

In contrast to a regular GA, the procedure in this section is based on the bi-population approach, as proposed by Debels and Vanhoucke (2005) for the RCPSP. This bi-population genetic algorithm (BPGA) makes use of two different populations: a population POP_R that only contains right-justified schedules and a population POP_L that only contains left-justified schedules. Both populations have the same population size POP. The procedure starts with the generation of an initial population of solution vectors. The forward-procedure is used to feed the population POP_L of left-justified schedules. This population is evaluated and the crossover and mutation operators are applied on the solution vectors in this population. The iterative forward-backward procedure of Li and Willis (1992) is applied: the backward-procedure is used to feed the population of right-justified schedules POP_R with combinations of population elements of POP_L that are scheduled backwards with the serial SGS. The forward-procedure is applied on the population elements of POP_R for updating POP_L. This process is repeated until the stop criterion is met. A conceptual overview of this approach is given in figure 3.1.

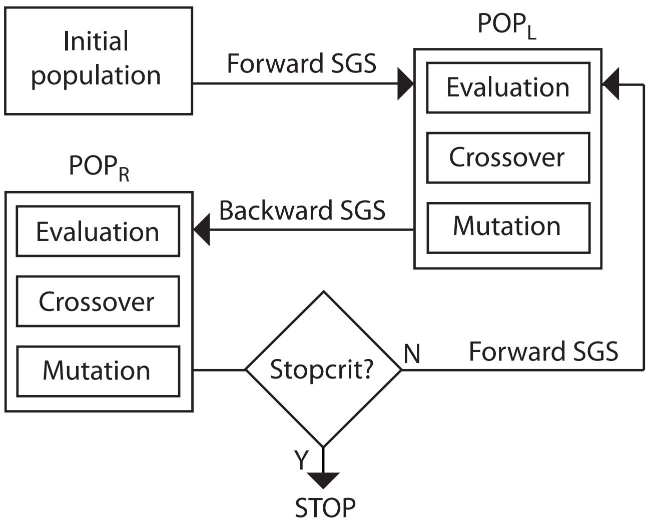


Figure 3.1: Procedure of the bi-population genetic algorithm

In the remainder of this section, we first examine the representation of the individuals, based on the random key representation with topological ordering notation proposed by Valls et al. (1999) (section 3.3.2). In section 3.3.3 the schedule generation scheme is described, which is extended with a local improvement search, adapted from Hartmann (2001). In section 3.3.4 the algorithmic details of the BPGA are proposed, while in section 3.3.5 the computational results are presented.

3.3.2 Representation

The representation of an individual is crucial for the performance of the genetic algorithm. Kolisch and Hartmann (1999) distinguish five different schedule representations in the RCPSP literature, from which the activity list (AL) representation and the random key (RK) representation are the most widespread. Hartmann and Kolisch (2000) conclude from experimental tests that procedures based on AL representations outperform the other procedures. However, Debels et al. (2006) illustrate that the unique RK representation also leads to promising results thanks to the use of the topological ordering (TO) notation (Valls et al., 1999). A topological order of the activities is an order which is compatible with the precedence relations of the project. It implies that for all activities i and j for which $s_i < s_j$, activity i should have a higher priority than activity j.

As the multi-mode is an extension of the single-mode RCPSP, most authors, such as Slowinski et al. (1994), Bouleimen and Lecocq (2003), Hartmann (2001), Józefowska et al. (2001) and Alcaraz et al. (2003), have used an extension of the standard AL representation. In this genetic algorithm procedure, however, we use the RK representation. An individual is therefore represented by a solution vector, which consists of a random key and a mode list.

Resume the example given in the previous chapter. Figure 3.2 shows a schedule with a makespan of 12 days. The schedule is based on the random key (1,2,3,4,5,6, 7,8) and the mode list (1,1,2,1,2,1,2,2). The schedule is feasible since the required nonrenewable resource units do not exceed the nonrenewable availability ($ERR =$ 0).

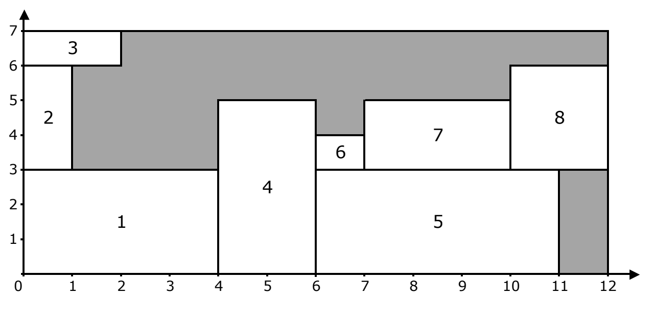


Figure 3.2: Schedule of the example project

3.3.3 Extended generation scheme

A schedule generation scheme (SGS) translates the solution vector into a schedule S. As already stated in section 3.2.1, we use in this work the serial SGS. However, we have extended the serial SGS with the following two elements.

First, our procedure makes use of the forward/backward scheduling technique, a well-known local search technique for RCPSP metaheuristics. The idea of scheduling activities in backward and forward direction has been introduced by Li and Willis (1992) and then used as improving method by Tormos and Lova (2001). Afterwards several authors, such as Alcaraz and Maroto (2001), Valls et al. (2005) and Debels et al. (2006) amongst others, have used similar procedures. The forward (backward) procedure is applied on the population POP_R (POP_L) of right-justified (left-justified) schedules and is used to build the left-justified (right-justified) schedules which are stored in the population POP_L (POP_R). To embed the TO condition in the random key representation, the start (finish) times of the activities of the schedules in POP_L (POP_R) are used as the priority values of the random key. The sequence of the activities is made by sorting the activities in (reverse) chronological order of their start (finish) times (Debels and Vanhoucke, 2007).

Second, our serial generation scheme is extended with a procedure to improve the mode selection. For every activity $i \in \{1, ..., |N|\}$ that will be scheduled, both in the forward and in the backward procedure, there is a probability of P_{modimp} that the mode improvement procedure will be executed. We use m_i to refer to the current mode of activity i. The mode improvement procedure evaluates for each activity i whether a new mode selection m'_i leads to an improvement in the ERR'. Consequently, this procedure evaluates all possible mode assignments $k = 1, ..., |M_i|$ of an activity i and calculates for each new mode list the corresponding excess of resource request ERR'. If the ERR' is equal or smaller than the current ERR, the procedure checks if an improvement can be made in the finish time of that activity. The mode m'_i with the lowest finish time f'_i, which does not increase the ERR, is chosen. If no improvement can be made, the mode selection is retained.

The pseudocode for the mode improvement procedure for an activity i of a project with an excess resource request of ERR is shown in table 3.2.

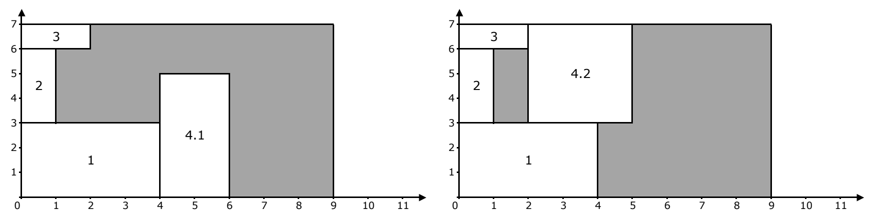
This mode improvement procedure is a combination of a mode selection rule of Lova et al. (2006) on the one hand and the single-pass improvement local search of Hartmann (2001) on the other hand. Lova et al. (2006) extend the serial generation scheme using the 'Earliest Feasible Finish Time' as a mode selection rule. During the generation of the schedule, this rule selects for each activity the execution mode so that it is scheduled with the smallest feasible finish time possible. The local search of Hartmann (2001) is based on the multi-mode left shift of Sprecher (1994). For each completely generated feasible schedule, the procedure checks for

```
FOR k=1,...,$|M_i|$
{
    Set new mode $m'_i = k$ and create a new mode vector
    Compute $ERR'$
    IF ($ERR'$)$\leq ERR$)
        Compute $f'_i$
        IF($f'_i < f_i$)
            $f_i = f'_i$
            $m_i = m'_i$
            $ERR = ERR'$
}
```

Table 3.2: Pseudocode of the mode optimization procedure for activity i

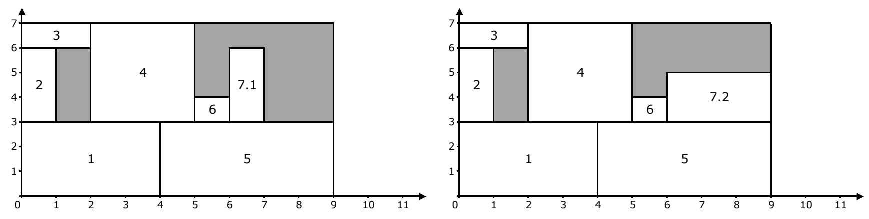
every activity whether a multi-mode left shift can be performed. A multi-mode left shift of an activity j is an operation on a given schedule which reduces the finish time of activity j without changing the modes or finish times of the other activities and without violating the constraints. Unlike the procedure of Hartmann, our procedure tries to improve the makespan during the generation of the schedule.

One could think that the mode improvement procedure will always chose the mode with the lowest duration. However, a mode with a shorter duration often requires more nonrenewable resource units and gives cause to an increase of the ERR and the infeasibility of the schedule. In addition, a mode with a longer duration requires less resource units, both renewable and nonrenewable. The procedure will therefore be able to schedule the activity more easily in parallel with other activities, because - due to the lower resource requirements - less resource conflicts will occur. Retake the schedule presented in figure 3.2. Suppose that activities 4 and 7 are subject to the mode improvement procedure. After scheduling activities 1, 2 and 3 (without applying the procedure), the mode improvement procedure is used for activity 4. With the current mode for activity 4 (mode 1), the schedule has an ERR of 0. The finish time of mode 1 is 6 (see Fig 3.3(a)). When changing the mode of activity 4 to 2, the ERR remains equal, because the required nonrenewable resource units (22) are smaller than the available ones (23). Although the duration of mode 2 is larger than mode 1, the required renewable resource units also decrease and the activity can be scheduled in parallel with activity 1. The finish time for mode 2 is now 5 (see Fig 3.3(b)). Because mode 2 has a lower finish time and the ERR does not deteriorate, the mode list is adapted. After scheduling activities 5 and 6, the mode improvement is applied on activity 7. For activity 7, mode 2 is currently selected and when inserting the activity in the current partial schedule, this activity has a finish time of 9 (see Fig 3.4(b)). However, when choosing mode 1, the finish time can be reduced to 7, because no renewable resource conflict occurs (see Fig 3.4(a)). The nonrenewable resource requirements increase to 23 and hence, the ERR remains 0. As the finish time is smaller for mode 1, the mode list for activity 7 is adapted to mode 1. After scheduling activity 8, the optimal solution for this specific problem is obtained, as shown in figure 3.5.



(a) Mode 1 (b) Mode 2

Figure 3.3: Mode improvement of activity 4



(a) Mode 1 (b) Mode 2

Figure 3.4: Mode improvement of activity 7

3.3.4 Details of the genetic algorithm

3.3.4.1 Initial population

The genetic algorithm is started by building an initial population POP_L of left-justified schedules. First, the random key is generated randomly. Second, for each activity i, $i \in \{1, ..., |N|\}$, an execution mode m_i is randomly selected. To minimize the number of infeasible solutions in the initial population, the local search procedure of Hartmann (2001) is applied to transform infeasible solutions into feasible ones. The procedure chooses an activity randomly and for that activity, a different mode is chosen. If the ERR remains the same or decreases, the mode for that activity is changed. This step is repeated until the mode assignment is feasible (ERR=0) or until J consecutive unsuccessful trials to improve the mode

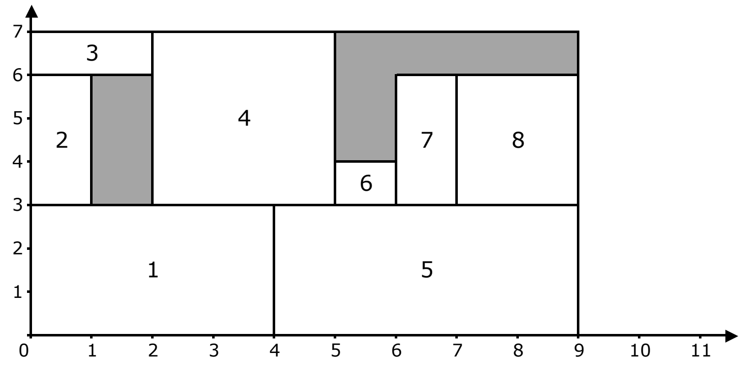


Figure 3.5: Optimal solution

assignment have been made. In this procedure, J equals to 4 times the number of activities in the project. Based on the random key and the mode list, a schedule is constructed with the extended serial generation scheme.

3.3.4.2 Evaluation

Once the initial population has been generated, each of the schedules must be evaluated. Therefore, a fitness value is calculated for each individual. In the single-mode RCPSP, the fitness value normally equals the makespan of the project C_{max}. It is a good measure for sequencing the different individuals according to their contribution to the objective of the problem. However, using the makespan as fitness value for the multi-mode RCPSP is inappropriate. As infeasible schedules (with an $ERR > 0$) can be included in the population, infeasible schedules must be penalized, otherwise they will displace the feasible schedules in the population. Józefowska et al. (2001) examine the differences between a fitness function with penalty function and one without and revealed that the fitness function with penalty function clearly performs better.

A good fitness function gives appropriate feedback to the genetic algorithm. Hartmann (2001) defines the fitness function for an individual as follows:

$$f_{HART} = \begin{cases} C_{max} & \text{if feasible} \\ T + ERR & \text{otherwise} \end{cases}$$

If the schedule is feasible ($ERR = 0$), the fitness function is equal to the makespan of the project C_{max}. If the schedule is infeasible, the fitness function is equal to the sum of the maximal durations of the activities (T) plus the ERR of the mode list. The lower the fitness value of a certain schedule is, the better the quality of the related schedule is. However, Alcaraz et al. (2003) point out that two different individuals with the same excess of resource request but with a different makespan have the same fitness value. They also mention that the upper bound T is a poor bound and indicate that the probability of the infeasible solutions to survive will be near 0. Therefore, they define a new fitness function that can be presented as follows:

$$f_{ALC} = \begin{cases} C_{max} & \text{if feasible} \\ C_{max} + max\_feas\_C_{max} - CP^{min} + ERR & \text{otherwise} \end{cases}$$

where $max\_feas\_C_{max}$ gives the maximal makespan of the feasible schedules related to individuals of the current generation and CP^{min} is the critical path using the minimal duration of each activity. According to Alcaraz et al. (2003), this fitness computation reveals better results than the one of Hartmann (2001).

Lova et al. (2009) also state that the fitness function of Alcaraz et al. (2003) is built by adding units of time from the makespan and units of resources from the excess of nonrenewable resources and conclude that the magnitude of both aspects of the solution can disturb the meaning of the fitness function. Therefore, they propose a new fitness function where both aspects of the solution are jointly considered but normalized in order to eliminate their magnitudes. This fitness function can be presented as follows:

$$f_{LOVA} = \begin{cases} 1 - \frac{max\_feas\_C_{max} - C_{max}}{max\_feas\_C_{max}} & \text{if feasible} \\ 1 + \frac{C_{max} - CP^{min}}{C_{max}} + \sum_{l \in R^{\nu}} max(0, \frac{\sum_{i=1}^{|N|} r^{\nu}_{im_il} - a^{\nu}_l}{a^{\nu}_l}) & \text{otherwise} \end{cases}$$

The individual with the greatest feasible makespan will have a fitness value equal to 1 while the individual with the best feasible makespan will have a fitness value close to 0. The fitness function of a non-feasible individual is always greater than 1, so feasible solutions will always have a smaller fitness value than infeasible solutions. Moreover, the sum of the normalized deviation of the makespan from the minimal critical path and the normalized excess of nonrenewable resources are added. According to Lova et al. (2009), this fitness function solves the cited weak points of the fitness computation of Alcaraz et al. (2003). In addition, computational tests in their paper reveal superior performance of this fitness function.

Finally, also Jarboui et al. (2008) propose a fitness function, which measures the degree of infeasibility and transforms this into a penalty function which grows in proportion to the infeasibility level, thus guiding the search toward the feasible solution space. The fitness function is given as:

$$f_{JARB} = \begin{cases} C_{max} & \text{if feasible} \\ C_{max} + \sum_{l=1}^{|R^{\nu}|} \delta \max(0, \sum_{i=1}^{|N|} (r^{\nu}_{im_il}) - a^{\nu}_l) & \text{otherwise} \end{cases}$$

with δ defined as a value greater than the maximal C_{max} of all solutions in order to inflate the value of the fitness, in case of infeasible solutions that induce their elimination.

During the computational experiments in section 3.3.5, the four fitness functions will be compared.

3.3.4.3 Parent selection

For each population element i of POP_L (POP_R) we create a set of 2 right-justified (left-justified) children that are candidates to enter POP_R (POP_L). To create a child out of i, another parent j from POP_L (POP_R) is selected by using the 2-tournament selection procedure. In this selection procedure two population-elements are chosen randomly and the element with the best fitness function value is selected. Afterwards, we determine randomly whether i or j represents the father. The other parent represents the mother.

3.3.4.4 Crossover

The crossover operation is applied on each pair of parents from POP_L (POP_R), producing two children which inherit parts of their characteristics. When a crossover is used, both the random key and mode list are adapted. We have tested different possible crossover operators. However, the so-called one-point crossover revealed the best results. In the one-point crossover, an integer number r is randomly selected. All data left from that point, both from the activity list and from the mode list, is copied from the father's chromosome. Beyond the point, the data of the mother is copied. Other crossover operators, such as the two-point crossover (Alcaraz et al., 2003), the uniform crossover (Özdamar, 1999) or the peak crossover (Valls et al., 2008; Debels et al., 2006), did not reveal better results than the one-point crossover.

3.3.4.5 Mutation

Mutation is applied to reintroduce lost genetic material into the population and creates variation in the different individuals. It introduces partial activity sequences and unselected mode choices into the population which could not have been produced by the crossover operator. In our genetic algorithm, two mutation operators are executed. The first one has a probability of P_{mut_act} and modifies the selected random key by randomly assigning a value between 0 and the start time of the dummy end activity. The mode assignment is not affected. The second mutation operator modifies the mode list with a probability of P_{mut_mod}. The mutation in both the random key and mode list is done randomly. Computational tests have revealed that the probabilities P_{mut_act} and P_{mut_mod} to obtain the best results are equal to 4% and 2%, respectively.

3.3.4.6 Update

In our algorithm, the parent population is replaced by the offspring population, which means that the population size remains the same. For the replacement, we follow the survival-of-the-fittest strategy. For every individual i out of the population, a partner is chosen (see 3.3.4.3) and 2 children are generated. The child with the lowest fitness value is selected and replaces individual i in the population, even if there is a deterioration. However, to avoid loosening high-quality schedules, we do not perform a replacement if the individual corresponds to a schedule with the best makespan found so far.

3.3.5 Computational results

In this section, the parameters of our genetic algorithm are configured on a training set with projects of 20 activities, each with three modes and 2 renewable and 2 nonrenewable resources. The order strength (OS) equals 0.25, 0.50 or 0.75. The renewable and nonrenewable resource strength equals 0.25, 0.50 or 0.75 and the resource factor equals 0.5 or 1. Using 5 instances for each problem class, we obtain a problem set with 540 network instances. For the generation of the instances, we have used the RanGen project scheduling instances generator developed by Vanhoucke et al. (2008) and extended the projects to a multi-mode version. On this dataset, we test the mode improvement procedure, the introduction of two populations and the different fitness computations. In table 3.3 the percentage deviation from the best found solution is shown. The following remarks can be made:

Mode improvement procedure The probability of applying the mode improvement procedure P_{modimp} is varying between 0% and 100% in steps of 10%. If we compare the results of the algorithm with and without the mode improvement procedure, the table shows that the introduction of the mode improvement procedure ameliorates the solution quality significantly.

Number of populations Although a bi-populational genetic algorithm is used, we also check if the one-populational genetic algorithm reveals other results. As shown in table 3.3, the best solution of the bi-populational GA clearly outperforms the best solution of the one-populational GA. In addition, statistical tests reveal a very significant ($p < 0.001$) contribution of the introduction of two populations in the genetic algorithm.

Fitness computation The different fitness functions are also compared. As we have mentioned in section 3.3.4.2, four fitness functions were proposed in the literature, i.e. the fitness function of Hartmann (2001), Alcaraz et al.

(2003), Lova et al. (2009) and Jarboui et al. (2008). Looking for the best results for both fitness computations, the fitness value of Alcaraz et al. (2003) reveals better results than the other fitness functions.

We thus conclude that the best results are obtained by using two populations, the fitness function according to Alcaraz et al. (2003) and setting the probability of applying the mode improvement procedure, P_{modimp}, equal to 30%.

When configuring the algorithm, we have noticed that the population size has an influence on the performance of the algorithm. We examined that the population size is negatively related to the number of activities. Similar results were found in Debels and Vanhoucke (2005), Hartmann (2001) and Alcaraz and Maroto (2001), amongst others. Tests were performed on different datasets with a different number of activities. For each dataset, the optimal population size was determined. The number of generated schedules was held constant at 5,000 schedules. A nonlinear least squares regression based on the best population size values for the different number of activities revealed a negative relationship between the population size (POP) and the number of activities ($|N|$) which can be defined as follows:

$$POP = e^{3.551 + \frac{22.72}{|N|}}$$

A large population size avoids homogeneity of the population and this becomes more important for small problem instances. However, when the population size becomes too large, only few generations can be computed within the time limit and the advantages of the genetic algorithm cannot be fully exploited.

3.3.6 Conclusions

In this section, a genetic algorithm for the MRCPSP has been proposed. For this algorithm we have used two populations, one with left-justified schedules and one with right-justified schedules. Computational tests have shown that this method performs better than the regular GA with one population. In addition, an extended serial schedule generation scheme is introduced, which improves the mode selection by choosing the feasible mode of a certain activity that minimizes the finish time of the activity. Finally, tests revealed that the fitness function of Alcaraz et al. (2003) performed better than the other ones available in the literature. The performance of this metaheuristic solution procedure on the current benchmark datasets will be shown in chapter 5.

Table 3.3: Results for the training set - configuration of the algorithm - average % deviation critical path length - 5,000 schedules

		P_{modimp}										
# populations	Fitness comp.	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
1 population	Hartmann	8.22%	1.83%	1.53%	1.49%	**1.42%**	1.50%	1.48%	1.56%	1.62%	1.59%	2.00%
	Alcaraz	7.93%	1.78%	1.51%	1.49%	**1.42%**	1.60%	1.43%	1.49%	1.50%	1.71%	2.04%
	Lova	8.19%	1.83%	1.53%	1.49%	**1.42%**	1.50%	1.48%	1.56%	1.62%	1.58%	1.98%
	Jarboui	7.23%	1.91%	1.68%	**1.46%**	1.48%	1.57%	1.70%	1.85%	1.81%	2.06%	2.53%
2 populations	Hartmann	1.03%	0.13%	0.07%	0.09%	0.06%	**0.03%**	0.17	0.09%	0.09%	0.04	0.08%
	Alcaraz	1.06%	0.13%	0.02%	**0.00%**	0.11%	0.06%	0.03%	0.11%	0.06%	0.01%	0.15%
	Lova	2.07%	0.49%	0.41%	0.38%	0.27%	0.35%	**0.16%**	0.28%	0.17%	0.22%	0.26%
	Jarboui	5.22%	2.03%	1.18%	0.94%	0.75%	0.51%	0.43%	0.40%	0.39%	0.35%	**0.31%**

3.4 Artificial Immune System

3.4.1 Introduction

An Artificial Immune System (AIS) is a computational algorithm proposed by De Castro and Timmis (2002) and inspired by theories and components of the vertebrate immune system. The vertebrate immune system is able to identify and kill disease-causing elements, called antigens, by the use of immune cells, of which the B-cells are the most common ones. These immune cells have receptor molecules on their surfaces (also called antibodies), whose aim it is to recognize and bind to pattern-specific antigens.

Since the antibodies on the B-cells are able to kill this specific type of antigens (antibodies and antigens whose shapes are complementary will rivet together), the B-cells will be stimulated to proliferate and to mature into non-dividing antibody secreting cells (plasma cells), according to the principles of clonal selection. The degree of proliferation is directly proportional to the recognizing degree of the antigen and the proliferation is succeeded by cell divisions and results in a population of clones which are copies from each other (De Castro and Timmis, 2002).

To better recognize the antigens, a whole mutation and selection process, which is called the affinity maturation, is applied on the cloned cells. A first mechanism is the hypermutation, a process in which random changes take place in the variable region of the antibody molecules. The degree of hypermutation is inversely proportional to the affinity of the antibody to the antigen: the higher the affinity, the lower the mutation rate and vice versa. However, a large proportion of these mutated antibodies will be of inferior quality and will be non-functional in the immune system. Those cells are eliminated from the population and replaced by newly developed receptors in a process called receptor editing.

The remainder of this section is organized as follows: the algorithmic details of the AIS algorithm will be presented in section 3.4.2. In section 3.4.3 some computational experiments will be executed, while in section 3.4.4 the conclusions are presented.

3.4.2 AIS algorithm for the MRCPSP

The efficient mechanisms of an immune system make artificial immune systems useful for scheduling problems. AIS is used for solving job-shop (Coello et al., 2003; Hart et al., 1998), flow-shop (Engin and Döyen, 2004) and resource-constrained project scheduling problems (Agarwal et al., 2007). In this section, a problem-solving technique for the MRCPSP based on the principles of the vertebrate immune system is presented. The different generic steps in our AIS algorithm are presented in figure 3.6 and will be discussed along the following subsections.

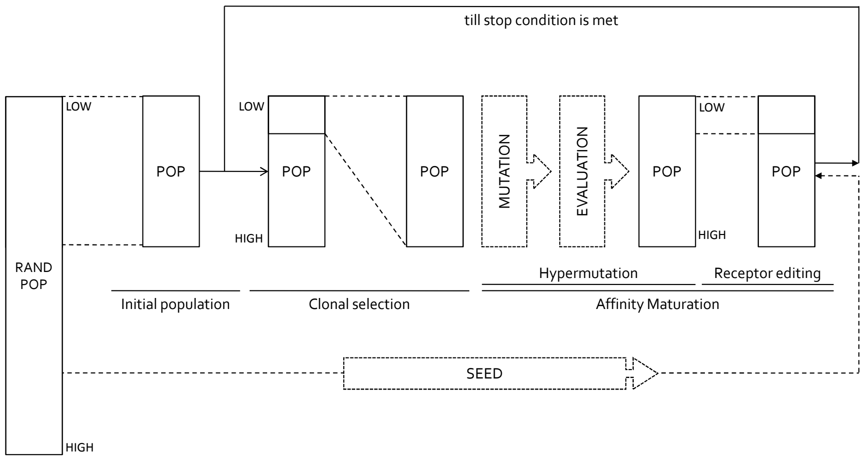


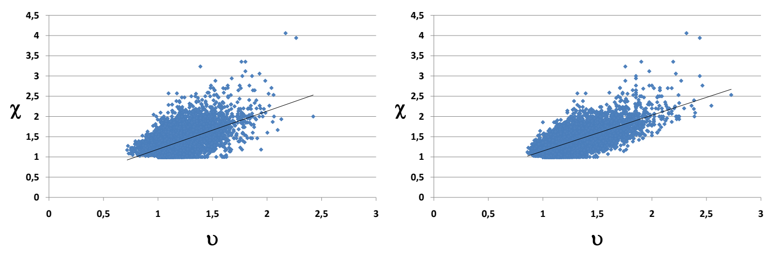
Figure 3.6: Artificial Immune System: procedure

3.4.2.1 Initial population

Several authors use random techniques to initiate the initial population. They argue that random initial starting solutions are more diverse and use less computational effort than heuristic procedures to produce initial solutions (Anderson and Ferris, 1994). However, in this work, we use a more controlled generation of the initial population. In a first stage, the mode list is generated, while in a second stage, the activity list is constructed based on this mode list.

Mode list generation A controlled mode assignment procedure is used to generate the mode lists of the initial population, and this approach will be compared with a random initial mode list population in the computational results sections. This procedure uses information obtained by a simple computational experiment performed on 90 project instances, randomly chosen out of the dataset as proposed in section 3.3.5. For each problem instance, 100 unique and feasible mode lists were generated, leading to 9,000 different combinations. For each combination, the near-optimal project makespan was calculated based on the bi-population genetic algorithm of Debels and Vanhoucke (2005).

The quality of all solutions was evaluated and compared with the characteristics of the generated mode lists. More precisely, various measures have been calculated to characterize the mode lists, such as the sum of all activity durations, the sum of the total work content, the average work content per resource and many more. In the remainder of this section, the focus is on the following two characteristics:



(a) Total work content (b) Sum of durations

Figure 3.7: Relationship between makespan and value of the mode assignment characteristic

- **Total Work Content** (TWC), defined as $\sum_{i=1}^{n} \sum_{k=1}^{|R^{\rho}|} r^{\rho}_{im_ik} d_{im_i}$
- **Sum of Durations** (SUM), defined as $\sum_{i=1}^{n} d_{im_i}$

A statistical analysis investigated the relation between the mode lists characteristics and the project makespans obtained during the experiment. In figure 3.7, a graph is plotted with the results of these tests. To standardize the values on the vertical and horizontal axis, the following calculations were made:

- On the vertical axis, χ is shown. For each problem instance and for each generated mode list, χ is calculated as the ratio of the makespan resulting from a particular mode assignment to the best found makespan. The larger χ, the more the makespan deviates from the best found makespan of that problem instance. The χ-value of the mode list which results in the best makespan is 1 and obviously, the value of χ can never be lower than 1.

- On the horizontal axis, the value υ is shown. υ is calculated as the ratio of the characteristic measure of the mode assignment to the characteristic measure of the mode assignment which results in the schedule with the best found makespan. υ can be lower than 1 since the value of the mode assignment characteristic with a higher makespan can be lower than the value of the mode assignment characteristic of the best makespan.

In both graphs a positive relationship between the makespan ratio and the ratio of the characteristic measure can be observed. A Pearson correlation test revealed that the correlations for the TWC and SUM are 0.61 and 0.74, respectively.

Since there is a significant and proportional relationship between the mode assignment measure and the makespan, mode lists with low measure values are preferred to mode lists with high measure values. This information was used during the generation of the initial mode lists population, leading to the controlled mode assignment procedure presented hereunder.

A *controlled mode assignment procedure* can be formulated in the three following steps:

1. A start population of mode lists, called $RANDPOP$, is created, with a large number of randomly generated feasible mode lists ($|RANDPOP|$ is equal to 4 times the number of populations elements POP).
2. Each mode list has a characteristic measure value, which is likely to lead to smaller project makespans.
3. The POP mode lists with the lowest values are selected for entrance in the initial population.

Once the mode lists are generated, a duration and resource consumption is set to each activity for each population element. Based on this information, the activity list can be generated.

Activity list generation In the second stage, the activity list is generated. Several heuristics available in the literature make use of well-performing priority rules to generate the sequence in which the activities should be scheduled. Some of the best performing priority rules are the *Latest Finish Time (LFT)*, the *Latest Start Time (LST)*, the *Minimum Slack (SLK)* and the *Maximum Remaining Work (RWK)*. More information about these priority rules can be found in Kolisch (1996a) and Boctor (1993).

For each mode list, an activity list is generated using one of the proposed priority rules. Since different mode lists are generated, also a diverse set of activity lists will be generated. If two or more equal mode lists occur in the initial population, different priority rules are applied on the mode lists to avoid equal solution vectors in the initial population.

After the generation of the activity list and the mode list, a schedule is built for each of the POP population elements in the initial population.

3.4.2.2 Clonal selection process

A population of *POP* solution vectors is generated and for each solution vector the makespan is determined. Only the best $P_{clonal}\%$ solution vectors will be available for proliferation. For these solution vectors, the corresponding affinity value is determined as follows:

$$Aff(V) = \frac{1}{makespan(V) - bestmakespan + 1} \tag{3.1}$$

where *makespan(V)* refers to the makespan of solution vector V and *bestmakespan* refers to the best makespan found so far. The number of clones of a solution vector V in the population is given by the affinity of the solution vector V over the sum of affinities of all population solution vectors multiplied by the number of population elements and can be formulated as follows:

$$\#clones(V) = \frac{Aff(V).POP}{\sum_{i=1}^{POP} Aff(V_i)} \tag{3.2}$$

Since the cloning of the antibodies is done directly proportional to their affinity value, it can be noticed that solution vectors with higher makespans will appear less frequently than solution vectors with low makespans. The size of the antibody population is fixed and infeasible solution vectors cannot be proliferated.

3.4.2.3 Affinity maturation

After the proliferation, the affinity maturation is performed. This process is applied in two phases: first, a hypermutation procedure is applied on each solution vector of the population and afterwards the receptor editing mechanism is used.

Hypermutation Since a solution vector contains both an activity list and a mode list, a mutation process is applied on both lists. However, the process for both lists is different.

For the mutation on the *activity list*, a hypermutation rate η which defines the degree of modification in the activity list is calculated. The hypermutation rate for solution vector V can be formulated as follows:

$$\eta(V) = 100e^{-0.05*(makespan(V)-bestmakespan)} \tag{3.3}$$

The number of mutations is calculated as:

$$NumbMut(V) = 1 + \frac{(100 - \eta(V))n}{100} \tag{3.4}$$

The lower the makespan, the lower the hypermutation rate of the activity list will be and the lower the number of mutations. By applying this mutation process, the algorithm can explore the neighborhood of the solution. That neighborhood will expand when the hypermutation rate increases. A mutation is defined as follows: in the current activity list, an activity is chosen and is moved randomly to a new position. In order to respect the precedence constraints, the new position of the activity is lying between the position of its latest predecessor and the position of its earliest successor (Hartmann, 2002).

The mutation process for the *mode list* is based on a frequency matrix of the mode lists in the population. To assign a mode to an activity, a random population element is chosen and the mode for that population element activity is assigned to the cloned element activity. Modes that occur more in the population of good solutions will therefore occur more. This procedure shows similarities with the Harmony Search procedure (Geem et al., 2001). In case the mode list becomes infeasible, a *feasibility procedure* is applied so that the mode list becomes feasible by changing mode assignments and the modes with the lowest mode characteristics (as defined in the previous section) are chosen first. After the mutation process, every new feasible solution vector is evaluated.

Receptor Editing After the cloning and mutation process, a new population of antibodies is generated. Only the best $P_{editing}\%$ antibodies are preserved in the population. The other elements are eliminated and to preserve POP elements in the population, the population is seeded with high quality mode lists from the start population $RANDPOP$. In that way, schedules with a low makespan stay incorporated in the antibody population and antibodies evolving to inferior search regions are deleted. The newly generated population is the start population for a new generation process. This process continues until the stop condition is met.

3.4.3 Computational results

In order to prove the efficiency of our algorithm, the AIS solution procedure is tested on the test set as proposed in section 3.3.5. The impact of the different algorithmic parameters, such as P_{clonal}, $P_{editing}$ and the population size POP, is tested and the efficiency of our initial population generation method is proven.

Extensive testing revealed that the optimal values for the different algorithmic parameters are P_{clonal} = 25%, $P_{editing}$ = 20% and POP = 450. Table 3.4 shows the average deviation from the minimal critical path *after the initial population generation* of POP population elements and *after 5,000 schedules*. The 2 different mode characteristics (TWC and SUM) and the 4 different priority rules (LFT, LST, SLK and RWK) are compared to the result of the solutions in which a random generated initial population is used (i.e. random generation of the activity list and random choice of the mode assignment).

A paired-samples T-test revealed a very significant ($p<0.001$) influence on the quality of the schedules when using a controlled initial population generation method instead of a random generation method: the difference between the average deviation from the critical path for the random generated solutions (48.23%) and the average deviation for the controlled generated elements (LST/SUM with an average deviation of 37.17%) is 11.06%, which corresponds with an average decrease of 2.2 working days on an average makespan of 28 days (*information not available in table 3.4*).

Regardless of which priority rule is used, the combination in which SUM

	After initial generation			**After 5,000 schedules**		
	Mode characteristic			*Mode characteristic*		
Priority rules	Random	TWC	SUM	Random	TWC	SUM
Random	48.23%	42.60%	42.43%	22.42%	20.81%	20.52%
LFT	43.24%	38.32%	37.85%	22.14%	20.86%	20.55%
LST	42.35%	37.62%	**37.17%**	22.44%	20.97%	20.74%
SLK	44.86%	39.56%	39.13%	22.29%	21.06%	20.73%
RWK	42.52%	37.95%	37.51%	22.22%	20.95%	**20.50%**

Table 3.4: Average % deviation from minimal critical path

is used as mode characteristic always outperforms the other mode characteristic TWC. This result is in accordance to the results of Boctor (1993), who proposed several priority rules for the mode assignment problem and who concluded that choosing the shortest feasible execution-mode is the most appropriate rule to minimize project duration.

The best solution after 5,000 schedules is found for the combination RWK and SUM (20.50%). There is a significant difference ($p<0.01$) with the solution in which a random generated initial population is used. During the tests performed in chapter 5, the *Maximum Remaining Work (RWK)* will be used as priority rule and the *Sum of Durations (SUM)* will be used as mode characteristic in our AIS algorithm.

3.4.4 Conclusions

In this section, an artificial immune system is presented. The vertebrate immune system mechanisms, which inspire the AIS solution methodology, were used to solve the MRCPSP. To generate a good and diverse initial population, a controlled search procedure is designed, which is based on an observed link between predefined mode list characteristic measures and the makespan of the mode assignment and which leads the search process more quickly to more interesting search regions. Moreover, it is remarkable that the AIS algorithm only makes use of a (hyper)mutation operator. The performance of the proposed AIS algorithm on the benchmark datasets will be tested in chapter 5.

3.5 Scatter Search

3.5.1 Introduction

Different research papers have provided valuable insights in the relation between project characteristics and the performance of solution procedures for both the single-mode and multi-mode RCPSP (see Herroelen and De Reyck, 1999; Kolisch et al., 1995). To the best of our knowledge, for the multi-mode RCPSP only one paper has used project characteristics to steer the algorithmic procedures towards promising solution areas. Buddhakulsomsiri and Kim (2007) propose the moving resource strength, which helps the priority rule-based heuristic for the MRCPSP with activity splitting and resource vacation in order to identify in which project situations activity splitting is likely to be beneficial during scheduling.

The main contribution of the solution procedure is threefold: first, a scatter search procedure to solve the scheduling problem is proposed. While a scatter search is proven to be successful in dealing with combinatorial problems, to the best of our knowledge, it has not been used before to solve the MRCPSP. Second, three different solution improvement methods are developed, each based on the information obtained from the renewable and nonrenewable resource characteristics. Third, the proposed algorithm provides state-of-the-art results for the available benchmark datasets.

The outline of this section is as follows. In section 3.5.2, a resource scarceness matrix is presented which gives insight into the influence of the scarceness of the renewable and nonrenewable resources on the search focus of the algorithm. Section 3.5.3 proposes a scatter search for the MRCPSP, for which different solution improvement methods tailored to the resource scarceness characteristics of a project, are presented. In section 3.5.4, computational results for the configuration of the scatter search are given and the influence of the resource scarceness on the solution quality is tested. Finally, in the last section, overall conclusions of this solution procedure are presented.

3.5.2 Resource scarceness matrix

Several resource parameters have been introduced in the past decades to measure the scarceness of resources of a project instance. These parameters are determined by the resource consumption and the resource availability. For a constant resource availability, the scarceness will increase for an increasing resource consumption. Moreover, the more restrictive a resource type becomes, the higher the makespan of the project will be.

Since the resource scarceness can be applied on both the renewable and nonrenewable resources, a brief description for both resource types is given.

- **Renewable resources**

  When the scarceness of the renewable resources is low, the project is hardly restricted by its resources. The makespan of the project will mainly be determined by the precedence relations of the project and each activity will be scheduled at or close to its critical path start time. When, however, the scarceness of renewable resources is high, the influence of the resource constraints will overrule the precedence constraints. Due to the relatively low resource availability, most of the activities will be scheduled one after the other. An increase from a low to a high resource scarceness also leads to an increasing deviation of the project makespan above the minimal critical path duration.

The renewable resource scarceness might influence the performance of a solution procedure. Projects with a low scarceness might need a search procedure which focuses on the neighborhood of the minimal critical path mode assignment (i.e. the critical path using the minimal duration of activities), while a search procedure that will mainly focus on the limitations imposed by the renewable resource availabilities might be more effective for projects with a high scarceness of the renewable resources.

– **Nonrenewable resources**

The feasibility of a mode assignment is determined by the nonrenewable resource consumption. A mode combination is feasible if the sum of the requested nonrenewable resources is smaller than or equal to the nonrenewable resource availabilities (i.e. if ERR is equal to 0). In case the scarceness of the nonrenewable resources is low, many mode assignment combinations will be feasible. The more the scarceness of the nonrenewable resources increases, the more mode combinations will become infeasible. Consequently, the higher the scarceness of the nonrenewable resources, the more the search procedure should focus on the search for feasible mode assignments.

Combining the information of the resource scarceness of both the renewable and nonrenewable resources, a *resource scarceness matrix* can be presented, as shown in figure 3.8. On the horizontal axis the scarceness of the renewable resources, moving from low to high, is presented, while on the vertical axis the scarceness of the nonrenewable resources is shown. The matrix can be divided into four quadrants: in quadrant 1 and 2 the number of feasible modes (#feas.modes) is high (indicated as '$>>$'), while the amount of feasible modes is more limited ('$<<$') in quadrants 3 and 4, due to the high nonrenewable resource scarceness. In quadrant 1 and 3 the makespan (C_{max}) of the projects with a low renewable resource scarceness will be close to the critical path duration (CP), while the makespan of the projects with high resource scarceness values in quadrants 2 and 4 might deviate significantly from the minimal critical path duration.

In the next section, a scatter search heuristic and different improvement methods are presented. The improvement methods are based on the resource scarceness characteristics of each quadrant to increase the effectiveness of the search procedure in each of the quadrants.

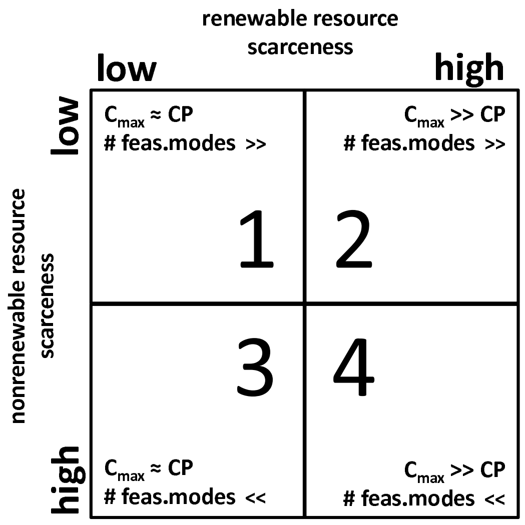


Figure 3.8: The resource scarceness matrix

3.5.3 Scatter search

Scatter search is a population-based metaheuristic procedure, proposed by Glover et al. (2000), in which solutions are intelligently combined to yield better solutions. The scatter search method involves deterministic procedures that can include problem specific knowledge (Pinol and Beasley, 2006) and can therefore be implemented in a variety of ways and degrees of sophistication. Scatter search algorithms are often classified as so-called evolutionary methods. However, the scatter search algorithm contrasts with other evolutionary procedures, such as genetic algorithms, by providing unifying principles of joining solutions based on generalized path constructions in Euclidian space and by utilizing strategic designs where other approaches resort to randomization (Glover et al., 2000).

For an overview of the basic and advanced features of the scatter search, we refer to Glover et al. (2000) and Marti et al. (2006). The scatter search we present in this work has a generic procedure as outlined in the pseudocode below.

```
1. Diversification Generation Method
While Stop Criterion not met
   2. Subset Generation Method
   3. Solution Combination Method
   4. Improvement Method
   5. Reference Set Update Method
Endwhile
```

In figure 3.9, a conceptual overview of the different steps in our scatter search procedure is shown. In the remainder of this section, each of these different steps is explained in detail.

3.5.3.1 The Diversification Generation Method

Initial population In this first step, a pool P of POP solution vectors is generated.

The controlled mode assignment procedure, as proposed in section 3.4.2.1, is used to generate the mode lists. Once the mode lists are generated, a duration and resource consumption can be assigned to each activity for each population element. Since Kolisch and Drexl (1997) mention that finding a feasible solution for the MRCPSP is a NP-complete problem if at least two nonrenewable resources are given, infeasible solutions are accepted in the initial population, but are penalized with the penalty function of Alcaraz et al. (2003). The activity lists are generated randomly, assigning a priority value to each activity.

Reference sets After the generation of POP population elements and the evaluation of the solution vectors with the serial schedule generation scheme (SGS), which translates the solution vector into a schedule, two diverse populations are constructed from P: a set B_1, with the b_1 best solutions of the solution set P and a set B_2, with b_2 diverse solutions. For the subset B_1, a threshold t_1 on the minimal distance between the elements is imposed in pursuit of diversity. The subset B_2 contains the b_2 best solutions from $P \backslash B_1$ that are sufficiently distant from the elements of B_1. The diversity in B_2 is achieved by a threshold t_2 on the smallest distance to any element in B_1 with $t_2 > t_1$. The distance between two solutions is a measure for diversity and is calculated according to the following two distance functions. The first distance function, $d^s_{p_1,p_2}$, calculates the distance as the sum of the differences between the start times of the activities and can be formulated as follows:

$$d^s_{p_1,p_2} = \sum_{i=1}^{|N|} |s_i^{p_2} - s_i^{p_1}|$$

with p_1 and p_2 two population elements and s^{p_1} and s^{p_2} their according start times. The second distance function, $d^m_{p_1,p_2}$, calculates the distance based on the difference in mode assignments and is formulated as follows:

$$d^m_{p_1,p_2} = \sum_{i=1}^{|N|} \begin{cases} 0 & \text{if } m_i^{p_1} = m_i^{p_2} \\ 1 & \text{otherwise} \end{cases}$$

In the computational results section both distance functions are compared and the optimal values for the thresholds are determined.

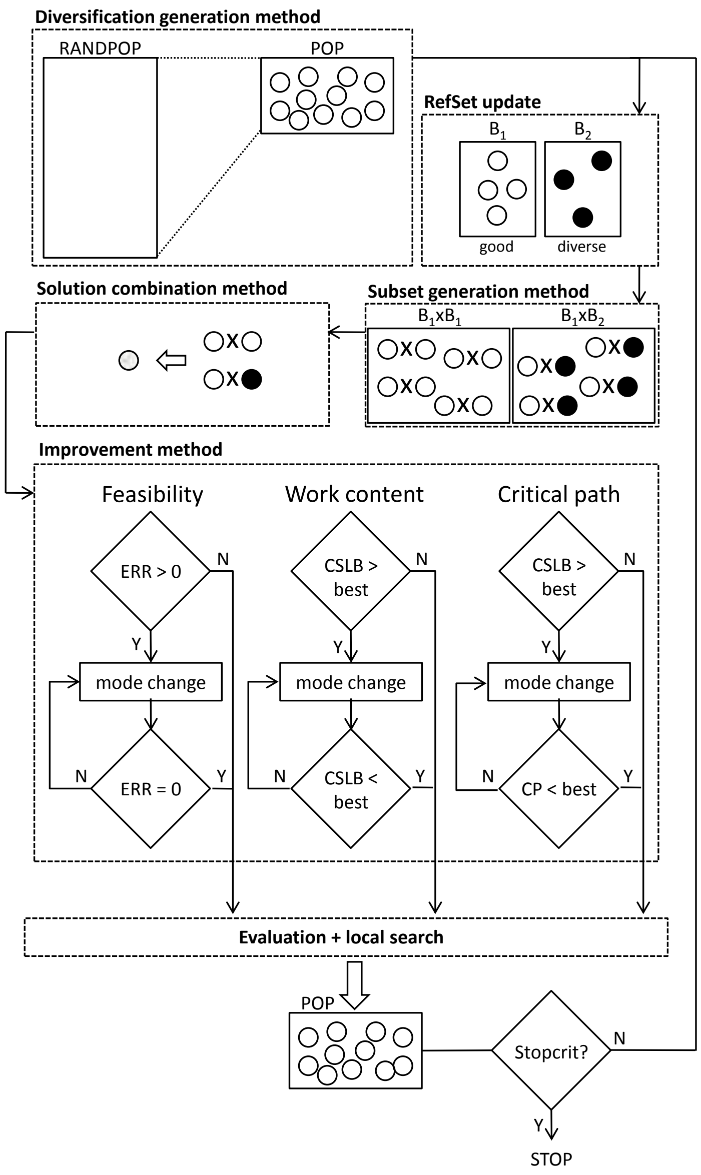


Figure 3.9: A conceptual overview of the scatter search procedure

If there are less solutions in B_2 than the predefined number b_2, the set B_2 is filled up with randomly generated schedules.

3.5.3.2 The Subset Generation Method

After the initialization phase, a new pool of solutions is created by combining pairs of reference solutions in a controlled way. New solutions are created from all two-element subsets. First, all pairs in B_1 containing at least one new solution compared to the previous generation are considered. From each such pair, two children are produced. Second, from each combination of one element from B_1 and one from B_2 two offsprings are constructed. Choosing the two reference solutions out of the same cluster stimulates intensification, while choosing them from different clusters stimulates diversification.

3.5.3.3 The Solution Combination Method

In the solution combination phase, the two selected population elements produce a new offspring which inherits parts of their parents' characteristics. Several crossover operators were tested and tests revealed that the two-point crossover method clearly outperforms other crossover operators. In the *two-point crossover scheme*, two crossover points are randomly chosen and the characteristics between them are exchanged. As the procedure works on both the activity list and the mode list, the crossover considers start times and modes simultaneously (i.e. using the same crossover points).

3.5.3.4 The Improvement Method

In this section, we propose different solution improvement methods, which are applied on the solution vectors that are generated in the solution combination method. Every new solution vector consists of a new generated activity list and a new generated mode list. Since the mode list determines the duration and the renewable and nonrenewable resource requirements for each activity, quick tests can be used to indicate whether the solution vector has potential to improve the current best solution found so far, without actually using the schedule generation scheme. The two quick tests are:

Feasibility test This test is related to the **nonrenewable resources** and checks whether the mode assignment is feasible or not. If the test reveals an infeasible solution vector ($ERR > 0$), the *feasibility improvement method* is performed.

Lower bound This test is related to the **renewable resources** and checks whether a new generated mode assignment (i.e. a duration and renewable resources) can lead to an improvement in the total project makespan. If the critical sequence lower bound (CSLB), proposed by Stinson et al. (1978), is larger than the best makespan found so far, two specific improvement methods are performed to improve the solution vector: the *critical path improvement method* tries to minimize the critical path length of the solution vector, while the *work content improvement method* tries to minimize the total work content of the proposed solution vector.

In case one of the these two tests is positive, one of the three improvement methods discussed below will be called in order to obtain a modified solution vector which will likely result in a decrease of the project makespan compared to the best known solution found so far. Consequently, each of the improvement methods will modify the mode list of the solution vector in such a way to maximize the probability that these modifications lead to a better project makespan. Therefore, a probability $p_{(i,j)}$ is calculated in order to determine which activity/mode combinations will be subject to change. The activity/mode combinations with a higher $p_{(i,j)}$ value will have a higher priority to be modified. The probability $p_{(i,j)}$ is defined as follows:

$$p_{(i,j)} = \frac{\Delta_{i,j}}{\sum_{i=1}^{N} \sum_{j=1}^{|M_i|} \Delta_{i,j}} \tag{3.5}$$

with $\Delta_{i,j}$ the *improvement value* for each activity i/mode j combination. Changes are made until a stop criterion defined by the improvement method is met.

Feasibility improvement method The purpose of this improvement method is to decrease the value of ERR. The improvement value $\Delta_{i,j}$ is formulated as follows:

$$\Delta_{i,j} = \max\{0, ERR_{old} - ERR_{new}\} \tag{3.6}$$

with ERR_{new} equal to the ERR-value based on the activity i/mode j combination, holding all other modes equal. Obviously, the value of ERR_{new} is equal to the value of ERR_{old} if the current mode m_i of activity i is chosen. Once the mode assignment becomes feasible or no further improvements can be made, the feasibility improvement method is stopped.

Critical path improvement method The purpose of this improvement method is to minimize the critical path length of the solution vector. The improvement value $\Delta_{i,j}$ for this improvement method is calculated as follows:

$$\Delta_{i,j} = \max\{0, CP_{old} - CP_{new}\} \tag{3.7}$$

with CP_{new} the critical path based on the duration of the activity i/mode j combination and holding all other modes equal. The improvement method stops when CP_{new} is smaller than the best found makespan or when no further improvements can be found.

Work content improvement method The purpose of this improvement method is to minimize the total work content of the proposed solution vector. The improvement value $\Delta_{i,j}$ for this improvement method is calculated as follows:

$$\Delta_{i,j} = \max\{0, WC_{old} - WC_{new}\} \tag{3.8}$$

with WC_{new} the needed work content based on the duration and resource demand of the selected mode, holding all other modes equal. The improvement method stops when the critical sequence lower bound is smaller than the best found makespan or when no further improvements can be found.

These improvement methods perfectly fit into the renewable and nonrenewable resource scarceness matrix presented in figure 3.8. Since the feasibility improvement method will try to solve nonrenewable resource infeasibilities expressed by positive ERR values, it will lay its focus on the third and fourth quadrants of the matrix. The focus of the critical path improvement method is to make changes in the critical path length, and hence lays its focus on the left part of the resource scarceness matrix. The work content improvement method puts a focus on the total work content of the activity/mode combinations, and consequently, will be fully exploited for project instances classified in the right part of the resource scarceness matrix. Figure 3.10 shows by means of the dark shaded areas what the focus of each improvement method is. The contribution of this approach on the solution quality will be tested in the computational results section.

3.5.3.5 The Reference Set Update Method

The population evolves over time with the entrance of new solution vectors and the removal of old solutions, searching to improve the quality of the best known solution. A new solution is introduced as a member in the reference set either if the solution vector has a better objective function value than the solution vector with the worst objective function value in B_1 or if the solution point is more diverse with respect to B_1 than the least diverse solution point in B_2.

3.5.3.6 Local searches

In section 3.5.3.4, we have proposed different solution improvement methods. These methods were applied on the infeasible solution vectors, before they were

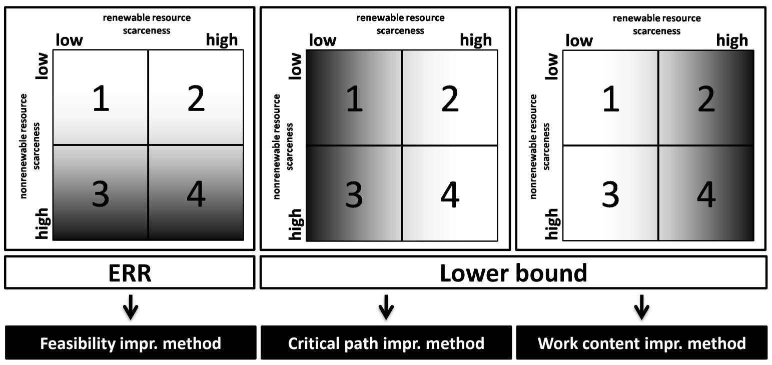

renewable resource scarceness
low
high
nonrenewable resource scarceness
low
high
1
2
3
4
ERR
Lower bound
Feasibility impr. method
Critical path impr. method
Work content impr. method

Figure 3.10: Solution improvement methods

actually scheduled, using the serial schedule generation scheme. However, different local searches applied on partially or fully scheduled projects were already proposed in the literature. The local search of Józefowska et al. (2001) and Bouleimen and Lecocq (2003) search in the neighborhood of a schedule by changing the activity list and mode list randomly. Kolisch and Drexl (1997) determine a probability function based on an approximation of the change in the objective function to determine which activity/mode pair will be changed. These local searches were coded and tested but revealed inferior results in the scatter search procedure with respect to the local search procedures of Hartmann (2001) and Van Peteghem and Vanhoucke (2010) (see section 3.5.4). In what follows, we explain both local search procedures briefly. Both local search procedures will be tested in section 3.5.4.

Hartmann (2001) The local search of Hartmann (2001) is based on the multi-mode left shift of Sprecher (1994). A multi-mode left shift of an activity j is an operation on a given schedule which reduces the finish time of activity j without changing the modes or finish times of the other activities and without violating the precedence and resource constraints. For each feasible schedule, the procedure checks for every activity whether a multi-mode left shift can be performed. For each activity, the first feasible multi-mode left shift is applied to the schedule. It is called a single-pass procedure, because every activity is considered only once for a multi-mode left shift.

Van Peteghem and Vanhoucke (2010) The local search of Van Peteghem and Vanhoucke (2010), as proposed in section 3.3.3, selects an activity with a certain probability and evaluates during the generation of a schedule all feasible mode assignments of the selected activity. For each new mode assignment the ERR is calculated. If the ERR is equal to or smaller than the current one, the local search checks if an improvement can be made in the finish time of that activity. The mode with the lowest finish time that does not increase the ERR is chosen.

3.5.4 Computational results

In this section, we configure the algorithm and evaluate its performance. In section 3.5.4.1 we present two datasets which are generated for this research and which will be used to test and configure the algorithmic parameter settings in section 3.5.4.2. The influence of the improvement methods on the different quadrants in the resource scarceness matrix is tested in section 3.5.4.3. The analysis of the local searches is presented in section 3.5.4.4, while the introduction of an integrated solution procedure is presented in section 3.5.4.5.

3.5.4.1 Dataset generation

In this section, two datasets are proposed, each containing a large set of data instances based on different complexity project parameters. A first dataset is used to configure the proposed scatter search algorithm, the other dataset is used to analyze the influence of the improvement methods and the local searches on the resource scarceness matrix.

For the generation of the instances of both datasets, we have used the RanGen project scheduling instances generator developed by Vanhoucke et al. (2008) and extended the projects to a multi-mode version. Each instance contains 50 activities, with three modes, two renewable resources and two nonrenewable resources. Dataset 1 is used in section 3.5.4.2 for the configuration of the algorithmic parameters, while dataset 2 is used in sections 3.5.4.3, 3.5.4.4 and 3.5.4.5 to analyze the performance of the improvement methods and local searches.

The following network and resource parameters were used for the two datasets. The values for each of these project characteristics are presented in table 3.5.

1. The network complexity is described by the *order strength*. In both datasets, the value of the OS is set at 0.25, 0.50 or 0.75.

2. In dataset 1, the resource strength is set at 0.25, 0.50 or 0.75, while in the other dataset, the parameter varies between 0 and 1 in steps of 0.10. The same parameter values are used for the nonrenewable resource strength in both datasets.

3. The *resource factor* (RF) indicates the percentage of resources required per activity and is set at 0.50 or 1.

4. For each problem class, 5 instances were generated. In the last row of table 3.5, the *total number of instances* generated is shown.

Dataset 1		Dataset 2	
OS	0.25-0.50-0.75	OS	0.25-0.50-0.75
RS^{ρ}	0.25-0.50-0.75	RS^{ρ}	0 to 1 (0.10)
RS^{ν}	0.25-0.50-0.75	RS^{ν}	0 to 1 (0.10)
RF	0.50-1.00	RF	0.50-1.00
#	540	#	7,260

Table 3.5: Parameter setting for the different datasets

3.5.4.2 Impact of the algorithmic parameters

In this section, dataset 1 is used to test the impact of the different parameters on the effectiveness of the procedure. The optimal size of the initial solution pool is tested and set to $|POP|$=7b, with b=b_1+b_2 and b_1=8 and b_2=7. In table 3.6 the *random mode assignment procedure* is compared with the *controlled mode assignment procedure* for each of the two distance functions $d^s_{p_1,p_2}$ and $d^m_{p_1,p_2}$. For each combination, the sum of the 540 projects makespans (shown in the column 'Sum') and the average deviation from the best solution procedure (shown in the column 'Dev.Best') is presented. As can be seen, the controlled mode assignment procedure combined with the distance function $d^s_{p_1,p_2}$ revealed the best results. Tests also revealed that the optimal values for t_1 and t_2 were 0.5$|N|$ and 1.5$|N|$, respectively, with $|N|$ the number of activities in the project.

3.5.4.3 Influence of the improvement method

In this section, the influence of the different improvement methods on the solution quality in the different quadrants of the resource scarceness matrix is tested. The results for the improvement methods applied on dataset 2 after 5,000 schedules and after 1 second are mentioned in table 3.6. The sum of all the project makespans is presented in the column 'Sum', while the column 'Dev.CP' (%) contains the

	Distance function			
	$d^s_{p_1,p_2}$		$d^m_{p_1,p_2}$	
Initial mode generation	Sum	Dev.Best	Sum	Dev.Best
Random	35,438	2.87%	35,593	3.74%
Controlled	**34,436**	**0.00%**	34,872	1.49%

Table 3.6: Influence of initial mode generation and distance functions

average deviation from the minimal critical path length. The column 'Dev.Best' contains the average deviation from the solutions of the best procedure, which will be presented later. Since the CPU-time needed to execute each of the improvement methods differs significantly (as can be seen in the column 'CPU-time'), computational tests were performed with a fixed time limit of 1 second in order to make a fair comparison. However, similar conclusions can be drawn.

In this section, we will consider the first 5 results. The other results will be discussed in the following sections.

Figure 3.11 shows the performance of each improvement method on all project instances of dataset 2 and confirms that the focus on each improvement corresponds with the resource scarceness characteristics as mentioned in section 3.5.3.4:

– The lower the renewable resource scarceness of a project instance is, the more effective the critical path improvement method is.

– The higher the renewable resource scarceness of a project instance is, the more effective the work content improvement method is.

– The higher the nonrenewable resource scarceness of a project instance is, the more effective the feasibility improvement method is.

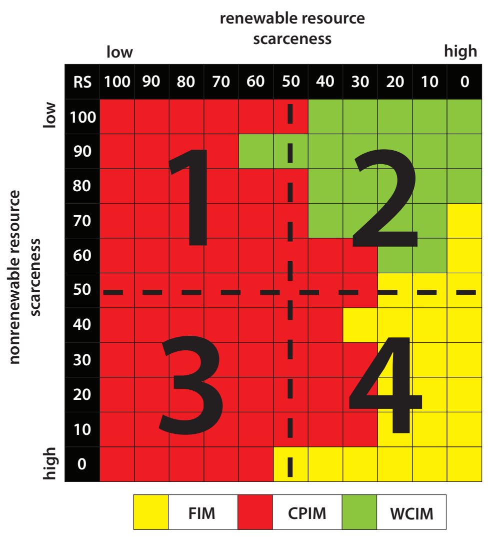


Figure 3.11: Distribution of the most effective improvement methods over the resource scarceness matrix

Based on these findings, a combined improvement method can be proposed. For this combined improvement method, a probability is assigned to each of the

Table 3.7: Influence of the improvement methods and local searches

Result	Impr. method	Local Search	5,000 schedules				1 second		
			Sum	Dev.CP (%)	Dev.Best (%)	CPU-time (s)	Sum	Dev.CP (%)	Dev.Best (%)
1	-	-	283,307	48.84%	4.29%	0.30	282,534	48.37%	4.37%
2	FIM	-	279,697	46.83%	2.95%	0.33	278,622	46.21%	2.90%
3	WCIM	-	279,872	46.92%	2.63%	0.67	279,257	46.55%	2.77%
4	CPIM	-	277,226	45.45%	1.33%	0.50	276,486	45.00%	1.41%
5	COMB	-	274,146	43.70%	0.42%	0.92	273,417	43.26%	0.57%
6	-	HART	274,494	43.90%	0.96%	0.25	272,518	42.69%	0.60%
7	-	VPV	273,869	43.60%	0.81%	0.29	272,209	42.58%	0.56%
8	-	COMB	273,104	43.21%	0.43%	0.26	271,876	42.38%	0.34%
9	COMB	COMB	272,547	42.81%	0.00%	0.72	271,369	42.09%	0.00%

improvement methods, based on the value U which is assigned according to the following formula:

$$U_{CPIM} = RS^{\rho}$$
$$U_{FIM} = 100 - RS^{\nu}$$
$$U_{WCIM} = 100 - RS^{\rho}$$

with RS^{ρ} the value of the renewable resource strength and RS^{ν} the value of the nonrenewable resource strength. The probability P that an improvement method q ($q = CPIM$, FIM or $WCIM$) is chosen is equal to:

$$P_q = \frac{U_q}{U_{CPIM} + U_{FIM} + U_{WCIM}}$$

If the improvement is selected and finds an improvement in the solution vector, the value U is increased so that the probability of selection in a next iteration is increased. In doing so, this probability selection method aims at selecting the improvement methods that fit best with the project under study given the renewable and nonrenewable resource characteristics as shown in the resource scarceness matrix. Table 3.7 shows that this method results in the best performing results found (result 5).

3.5.4.4 Introduction of local searches

Results 6 and 7 in table 3.7 show the impact on the solution quality for the two local searches described in section 3.5.3.6, i.e. the local search of Hartmann ($HART$) and the local search of Van Peteghem and Vanhoucke (VPV). The table shows that the local of search of Van Peteghem and Vanhoucke outperforms on average the solutions of Hartmann. However, looking to the specific $RS^{\nu} - RS^{\rho}$-combinations in the resource scarceness matrix, a distinction can be made: in (parts of) quadrants 2 and 3 the local search of Hartmann ($HART$) outperforms the local search of Van Peteghem and Vanhoucke (VPV), while in the other two quadrants, the latter seems more effective than the former (see fig. 3.12). Based on these findings, a controlled local search procedure is proposed (result 8), where in the specific regions of quadrant 2 and 3 the local search of Hartmann is used, while in the other situations, the local search of Van Peteghem and Vanhoucke is performed.

3.5.4.5 An integrated solution procedure for the MRCPSP

An integrated procedure is developed by integrating the combined local search and the combined improvement method in our scatter search algorithm. The results for the overall procedure are mentioned in table 3.7 (result 9). In the following section, tests will be performed on the PSPLIB dataset using this integrated solution procedure.

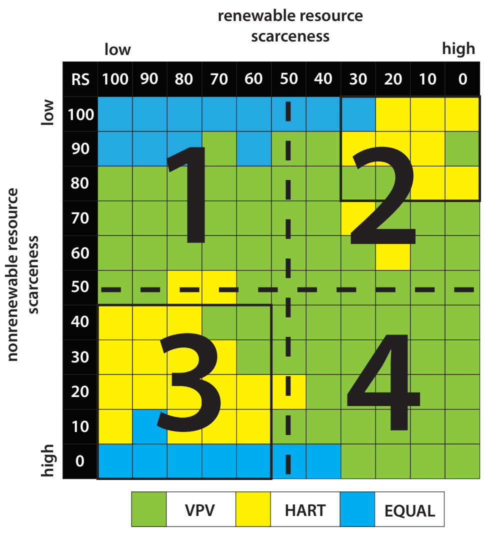


Figure 3.12: Influence of the local searches on the resource scarceness matrix

3.5.5 Conclusions

In this section, we have designed a promising scatter search procedure for the MRCPSP. The added value of this algorithm lies in the steering power of three proposed improvement methods, each tailored to the specific characteristics of different renewable and nonrenewable resource scarceness values. Computational results confirm the influence of these improvement methods on projects situated in one of the quadrants of our resource scarceness matrix. The combination of these improvement methods and the introduction of two local searches into one overall solution procedure leads to promising computational results.

3.6 Conclusions

In this chapter, we have presented three metaheuristic solution procedures for the MRCPSP: a genetic algorithm, an artificial immune system and a scatter search procedure.

The contribution of this chapter is threefold. First, three different population-based metaheuristic strategies are followed. The genetic algorithm uses several mechanisms such as natural selection, crossover and mutation in order to recombine existing solutions so that new ones are obtained. The scatter search algorithm contrasts with the genetic algorithm by focusing not only on the quality of the solution, but also on the diversity of a solution. The use of diversification and improvement methods leads to a more efficient exploration of the solution space. The artificial immune system finally makes use of mechanisms, such as the clonal selection process, hypermutation and receptor editing, which are inspired by the vertebrate immune system. The artificial immune system differs from the genetic algorithm by the fact that new solutions are only generated by applying a (hyper)mutation operator. Second, problem-specific techniques and local searches are added to each of the metaheuristic solution procedures in order to solve the MRCPSP efficiently: the extended generation scheme with mode improvement method in the genetic algorithm, the controlled mode assignment procedure to generate a good and diverse initial population in the artificial immune system procedure and the three improvement methods, each tailored to the specific renewable and nonrenewable scarceness characteristics in the scatter search procedure. The introduction of these methods significantly increases the performance of the procedures. Third, in the recent literature different penalty functions and local searches were proposed. An overview of these penalty functions and local searches is given and tested during computational experiments. The results give an indication of the effectiveness of each of these procedures and functions and emphasize the need for solution methods and procedures tailored to the specific characteristics of the multi-mode scheduling problem.

In this chapter, an overview is given of the algorithmic features of the different metaheuristic solution procedures. These procedures are tested on specific datasets to optimize the different algorithmic parameters, however, no comparison is made between the metaheuristics on a standard benchmark dataset. In order to make this fair comparison between the three procedures and the other metaheuristics available in the literature, computational tests are performed on the benchmark datasets PSPLIB and Boctor. The results of this comparison are presented in chapter 5. Moreover, results are also provided for tests performed on the new benchmark dataset MMLIB, which is presented in the next chapter.

4 New Dataset for the MRCPSP

4.1 Introduction

In order to make a fair comparison between different solution procedures for specific optimization problems, standard benchmark sets are available to test the efficiency and performance of the proposed algorithm. Examples are the PSPLIB dataset for the RCPSP (Kolisch et al., 1995), the MPSPLIB for multi-project scheduling problems (Homberger, 2007) and the dataset for the machine scheduling problem, proposed by Sels and Vanhoucke (2009).

For the MRCPSP, the PSPLIB dataset (Kolisch et al., 1995) and the Boctor dataset (Boctor, 1993) are available. Most authors make use of these datasets to test the performance of their procedures in order to compare them to other algorithms available in the literature. However, the two datasets show some important shortcomings given the recent evolution in the development of metaheuristic search procedures. Therefore, we propose in this chapter a new benchmark dataset which overcomes the disadvantages of the current datasets and which can be used as a benchmark dataset for further research. In section 4.2, we make an analysis of the current benchmark datasets and explain their shortcomings, while in section 4.3 we propose the indicators of the newly developed and generated benchmark dataset.

	PSPLIB	Boctor
Number of instances/dataset	640	120
Number of feasible instances/dataset	549 (avg.)	120
Number of activities	10, 12, 14, 16, 18, 20, 30	50, 100
Number of renewable resources	2	1, 2, 4
Number of nonrenewable resources	2	0
Number of modes	3	1,2,4

Table 4.1: Overview of the characteristics of PSPLIB and Boctor

4.2 Analysis of the current benchmark datasets

In this section, we make an analysis of the two current benchmark datasets, the PSPLIB dataset and the Boctor dataset. In table 4.1, an overview of the input parameters of both datasets is given.

PSPLIB The PSPLIB dataset is generated with the project generator ProGen (Kolisch et al., 1995) and is available in the project scheduling problem library PSPLIB from the ftp server of the Technische Universität München (http://129.187.106.231/psplib/). The dataset contains 7 subsets: the datasets J10, J12, J14, J16, J18, J20 and J30, containing project instances with 10, 12, 14, 16, 18, 20 and 30 activities, respectively. For each project, 2 renewable and 2 nonrenewable resources are used.

Boctor This dataset, proposed by Boctor (1993), contains 240 instances. There is one set composed of 120 instances of 50 activities (Boctor50) and one set composed of 120 problems of 100 activities (Boctor100). For each project, one, two or four renewable resource types are used.

Several project parameters have been introduced in the literature for describing the characteristics of a project network and the resource scarceness. The coefficient of network complexity (CNC, Pascoe, 1966), the order strength (OS, Mastor, 1970) and the I2 indicator (Vanhoucke et al., 2008) are examples of network topology measures. The resource factor (RF, Pascoe, 1966) and resource strength (RS, Cooper, 1976; Kolisch et al., 1995) are examples of resource scarceness measures.

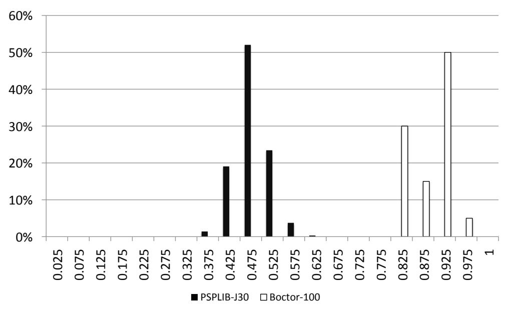
An overview of the values of these instance characteristics for the J30 and Boctor100 dataset is given in table 4.2. In this table, the average value, the minimum value and the maximum value for the different project characteristics (OS, CNC and I2) and the different resource characteristics (resource strength and resource factor) are presented. In figure 4.1, a frequency graph is given for the values of both the order strength and resource strength. As can be seen, the range for the values of the order strength (for J30 and Boctor100) and the resource strength (for Boctor100) is rather limited.

	J30			**Boctor100**		
	avg.	*min*	*max*	*avg.*	*min*	*max*
Project characteristics						
OS	0.46	0.34	0.61	0.87	0.79	0.93
CNC	1.81	1.81	1.81	1.59	1.44	1.92
I2	0.29	0.23	0.43	0.53	0.40	0.62
Resource characteristics						
Resource strength (R)	0.62	0.25	1	0.15	0.06	0.25
Resource factor (R)	0.75	0.5	1	0.88	0.67	1
Resource strength (NR)	0.62	0.25	1	-	-	-
Resource factor (NR)	0.75	0.5	1	-	-	-

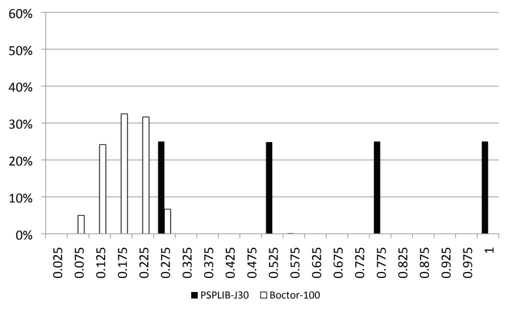
Table 4.2: Overview of the characteristics of PSPLIB and Boctor

Despite the diverse range of the resource parameters (the RS varies between 0.25 and 1) and the incorporation of both renewable and nonrenewable resources in the J30 dataset, we believe that the instances have four major shortcomings to stimulate further research for the MRCPSP. A first shortcoming lies in the inability to report feasible solutions for all problem instances. As an example, only 552 instances from the 640 generated instances have a possible feasible solution. A similar shortcoming holds for the J10 to J20 datasets (see Alcaraz et al., 2003; Van Peteghem and Vanhoucke, 2010). A second drawback is the small range of OS values (between 0.35 and 0.60) to guarantee a diverse set of project topology networks. A third reason why the current J30 instances are no longer desirable for further algorithmic development lies in the observation that most instances have been solved to near optimality, which leaves little room to reach major improvements with newly developed solution procedures. Future instances should contain projects with more activities (> 30), more mode combinations per activity (> 3) or more renewable or nonrenewable resources. Finally, the generation of activity modes should be done differently, in order to guarantee that all modes are efficient. In the J30 dataset, three modes have been generated per activity. However, a number of these modes can be deleted by the preprocessing procedure of Sprecher (2000), leading to only 2.88 modes per activity on average (see Van Peteghem and Vanhoucke, 2010). Moreover, the deletion of inefficient modes leads to a change of the resource parameters of the project instance, which could possibly lead to biased results.

The main advantage of the Boctor100 dataset is the large number of activities per project instance. However, three major concerns about this dataset can be specified. First, only renewable resources are taken into account and nonrenewable resources are neglected. Second, the average resource strength per project is not larger than 0.25, which means that the renewable resources are almost not restricted, and third, the order strength of the projects varies between 0.8 and 0.95,



(a) Order strength



(b) Resource strength

Figure 4.1: Frequency table for the project instance characteristics

which means that the projects are mainly serial.

In the next section, we will present three new datasets, which will cover most of the shortcomings mentioned in this section and are suggested to be used as a new benchmark dataset for the MRCPSP.

4.3 A New Dataset for the MRCPSP

4.3.1 Generation conditions

In order to overcome the shortcomings of the PSPLIB and Boctor datasets, the following conditions were taken into account while generating the dataset:

1. The generated project is diverse with respect to the project (OS) and resource characteristics (RS and RF).
2. Every instance has at least 1 feasible solution.
3. No modes can be excluded.
4. Both renewable and nonrenewable resources are taken into account.

4.3.2 Dataset generation

For the generation of the instances, we have used the RanGen project scheduling instances generator developed by Vanhoucke et al. (2008) and extended to projects with multiple modes. In order to meet the conditions mentioned in the previous section, two repair functions are used during the generation of the different datasets.

1. For projects with low values of the nonrenewable resource strength (RS^{ν} = 0.25), many infeasible projects were generated. This problem was also cited by Demeulemeester et al. (2003). If the resource strength of a nonrenewable resource is low, this results in a low value of the availability a^{ν}. Since the generation of the projects occurs randomly, only in a limited number of cases a feasible project is generated. Therefore, a repair function is used to avoid this problem. A large set of small projects (10 activities) is generated with a $RS^{\nu} = 0.25$ and a $RF^{\nu} = 0.50$. All feasible projects are selected from this large set. During the generation of the projects with a large number of activities (e.g. 50 or 100), a random combination is selected from the generated small subset of feasible projects and the nonrenewable resource information is randomly assigned to one of the new project's activities. To obtain the overall availability, the sum of all nonrenewable resource availabilities of the chosen small projects is made. This process can be executed since, based on the mathematical formulation of the nonrenewable resource strength, the combination of two feasible projects always leads to a feasible project.

2. Many inefficient modes were generated during the data generation process. Modes are labelled as *inefficient* if there is another mode of the same activity with the same or smaller duration and no more requirements for all resources. Since all projects are generated randomly, it is straightforward that inefficient modes are generated. Therefore, a repair function is used in order to deal with the problem of inefficient modes. If an inefficient mode is detected, the resource requirement of one of the nonrenewable resources is set to a value lower than the original nonrenewable resource demand. This is only possible if the resource demand is not 0 or if the nonrenewable resource demand is not the minimal resource demand for that activity (otherwise the nonrenewable resource strength will be affected). In the case no changes can be performed, a new project is generated.

4.3.3 Dataset characteristics

The dataset generation resulted in the design of two datasets: the MMLIB* dataset, which has project characteristics similar to the PSPLIB dataset and takes the shortcomings mentioned in the previous section into account, and the MMLIB+ dataset, which also takes more resources and more modes into account. The motivation to generate both datasets comes from the idea that the complexity of the current benchmark datasets should be increased. The MMLIB* is generated to increase the complexity of the scheduling problem (i.e. more activities), the MMLIB+ dataset is generated to increase the complexity of both the scheduling problem (i.e. more activities) and the mode assignment problem (i.e. more possible nonrenewable mode combinations).

MMLIB* The MMLIB* dataset is a dataset with two subsets: MMLIB50 and MMLIB100 with 50 and 100 activities, respectively. Each project activity has 2 renewable and 2 nonrenewable resources and for each activity, 3 modes were defined. The order strength is set at 0.25, 0.50 or 0.75. The resource strength of both the renewable and nonrenewable resource strength is set at 0.25, 0.50 or 0.75 and the resource factor is set at 0.50 or 1 for both renewable and nonrenewable resources. Using 5 instances for each problem class, each MMLIB* dataset contains 540 instances. The MMLIB* datasets can be considered as similar to the PSPLIB datasets, however, the order strength is more diverse and projects with a resource strength equal to 1 have not been included in the set. Moreover, all the instances have a feasible solution and no modes can be deleted during the preprocessing procedure.

MMLIB+ The MMLIB+ dataset contains projects with 50 and 100 activities. Each project activity has 2 or 4 renewable and nonrenewable resources. For each activity, 3, 6 or 9 modes are defined. The order strength is set at 0.25, 0.50 or 0.75. The resource strength of both the renewable and nonrenewable resource strength is set at 0.25, 0.50 or 0.75. In order to keep the number of instances to a reasonable level, the renewable and nonrenewable resource factor is set at 1. Using 5 instances for each problem class, the MMLIB+ dataset contains 3,240 instances.

4.3.4 Download

The problem instances, as well as the solution files, can be downloaded from the website http://www.projectmanagement.UGent.be/mrcpsp.htm, under the names MMLIB50.zip, MMLIB100.zip and MMLIB+.zip.

The solution files contain the best individual solution of each problem instance (without a given stop criterion). The website will be updated regularly, and we call upon researchers to report the solutions of their procedures when this leads to an improvement (with a stop criterion of 1,000, 5,000 and 50,000 schedules and without any stop criterion).

4.4 Conclusions

In this chapter, a new dataset for the MRCPSP is proposed which deals with the major shortcomings of the existing benchmark datasets and which will contribute to the search of new and better solutions in the future. In the next chapter, an objective comparison of various metaheuristic procedures on this new dataset will be presented.

5 Comparative Results for the MRCPSP

5.1 Introduction

The increasing interest in operations research for metaheuristics during the recent years has resulted in the development of several metaheuristic solution procedures for the MRCPSP. A wide variety of metaheuristic strategies, solution representations and schedule generation schemes were used to develop the most efficient algorithm. Since the methods are tested on different benchmark datasets using different stop criteria, a fair comparison between each of these procedures is difficult.

In this chapter, we compare all metaheuristic solution procedures available in the literature. We have coded all procedures as described in our literature overview presented in section 2.5.3 and compared their performance with the performance of our proposed solution procedures. This comparison gives an indication of the performance of our own procedures and classifies all procedures according to similar stop criteria. The tests are performed on the benchmark datasets PSPLIB and Boctor, as well as on the newly developed dataset MMLIB.

The remainder of this chapter is organized as follows. In section 5.2, an overview of the followed methodology to obtain the comparative results for the different metaheuristics is described. In section 5.3, the computational comparison is given, with a description of the test design and the results for the PPLIP, Boctor and MMLIB dataset. In the last section, overall conclusions and suggestions for future research are presented.

5.2 Methodology

In order to compare each of the procedures on the same computer and for the same stop criteria, each algorithm presented in section 2.5.3 has been coded[1]. In this section, we will explain the methodology that has been followed in order to obtain a fair and realistic reproduction of the original codes.

Each paper is studied by two undergraduate students and one PhD student. For each procedure three independently coded programs were available, each reviewed by a PhD student, who reviewed the code to look if the structure corresponded to the steps presented in the original paper. In case the content of the paper was insufficient to interpret the algorithm, other papers and solution procedures of the author were analyzed. The best program was selected and submitted to a second control phase.

If necessary, the following two adaptations were made. First, since not all algorithms included the preprocessing method of Sprecher et al. (1997), the preprocessing procedure was added to each solution procedure in order to make a fair comparison based on the same solution space. Second, procedures which are only applicable on datasets with renewable resources (in casu $BOCT$ and $MORI$), are transferred to a version in which nonrenewable resources are incorporated. In order to deal with possible infeasible solutions, the penalty function as introduced by Alcaraz et al. (2003) is used in the code.

To emphasize the efficiency of all procedures, an overview of the different solution procedures is given in table 5.1. For every procedure, the dataset and stop criterion is mentioned. The results in the paper are also indicated, as well as the results obtained by our own coded versions. As can be seen, all the procedures obtain equivalent or slightly better results than the results shown in the original papers. This can be due to randomness, the incorporation of the preprocessing process or the competitive nature of the coding process. The different procedures were tested on the available datasets as well as on the new dataset MMLIB.

5.3 Computational comparison

5.3.1 Test design

This section gives a comparison of all the metaheuristics presented in table 2.7 and the designed metaheuristic solution procedures proposed in chapter 3. Tests are executed on two subsets: a first subset with the existing datasets PSPLIB and Boctor (*J10*, *J20*, *J30* and *Boctor*) and a second subset with the newly generated datasets MMLIB* and MMLIB+.

[1]This overview contains all metaheuristic solution procedures published in international peer reviewed journals before August 1, 2009.

Table 5.1: Results of all procedures in the literature (original results and our coded results)

	Author	Year	Procedure	Dataset	Stop criterion	Result paper	Result analyse	Remarks
1	Slowinski et al.	1994	SA	Own	0.5 sec	0.02	0.02	
2	Boctor	1996	SA	Boctor50	0.5 sec	25.70	25.13	CP
				Boctor100	0.5 sec	27.20	26.82	CP
3	Mori and Tseng	1997	GA	-	-	-	-	
4	Özdamar	1999	GA	-	-	-	-	
5	Nonobe and Ibaraki	2001	TS	-	-	-	-	
6	Jozefowska et al.	2001	SA	J10	5,000 schedules	1.16	0.99	
				J20	5,000 schedules	6.74	6.65	
7	Hartmann	2001	GA	J10	6,000 schedules	0.10	0.13	
8	Bouleimen and Lecocq	2003	SA	J10	50 sec	0.21	0.18	
				J20	50 sec	2.10	2.09	
9	Alcaraz et al.	2003	GA	J10	5,000 schedules	0.24	0.24	
				J20	5,000 schedules	1.91	1.96	
				Boctor50	5,000 schedules	26.52	26.48	CP
				Boctor100	5,000 schedules	29.16	29.10	CP
10	Zhang et al.	2006	PS	J10	unknown/1 sec	0.11	0.12	
				J20	unknown/1 sec	1.79	2.41	
11	Jarboui et al.	2008	PS	J10	150ms	0.03	0.03	
				J20	150ms	1.10	1.08	
12	Ranjbar et al.	2008	SS	J10	5,000 schedules	0.18	0.17	
				J20	5,000 schedules	1.64	1.31	
13	Lova et al.	2009	GA	J10	5,000 schedules	0.06	0.04	
				J20	5,000 schedules	0.87	0.89	
				J30	5,000 schedules	14.77	14.58	CP
				Boctor50	5,000 schedules	23.70	23.65	CP
				Boctor100	5,000 schedules	24.85	24.63	CP
14	Tseng and Chen	2009	GA	J10	5,000 schedules	0.33	0.32	
				J20	5,000 schedules	1.71	1.47	
				J30	5,000 schedules	18.33	17.06	CP
15	Van Peteghem and Vanhoucke	2009	GA	J10	5,000 schedules	0.01	0.01	
				J20	5,000 schedules	0.57	0.57	
				J30	5,000 schedules	13.75	13.75	CP
				Boctor50	5,000 schedules	23.41	23.41	CP
				Boctor100	5,000 schedules	24.67	24.67	CP
16	Van Peteghem and Vanhoucke	2009	AIS	J10	5,000 schedules	0.02	0.02	
				J20	5,000 schedules	0.70	0.70	
17	Van Peteghem and Vanhoucke	2009	SS	J10	5,000 schedules	0.00	0.00	
				J20	5,000 schedules	0.32	0.32	
				J30	5,000 schedules	13.66	13.66	CP

CP = deviation from critical path

Since it is assumed that the computational effort for constructing one schedule is similar in most heuristics and in order to make a fair comparison, the evaluation is stopped after a predefined number of generated schedules. The evaluation of the first subset is stopped after 5,000 schedules. For the second dataset, the stop criterion is set at 1,000, 5,000 and 50,000 schedules.

According to Kolisch and Hartmann (2006) the advantage of the number of schedules as stop criterion is twofold: first, it is *platform independent* and second, future studies can easily make use of the *benchmark results* by applying the same stop criterion. However, the stop criterion also has a few shortcomings: first, it cannot be applied to all different heuristic strategies. Second, the required time to compute one schedule might differ between metaheuristics. Nevertheless, Kolisch and Hartmann (2006) conclude that limiting the number of schedules is the best criterion available for a broad comparison, which motivated us to use this stop criterion in all computational experiments.

To measure the number of schedules, the definition of one schedule should be defined. In their RCPSP review paper, Kolisch and Hartmann (2006) state that one schedule corresponds to (at most) one start time assignment per activity, as done by a SGS. However, measuring the number of schedules according to this rule means that for every mode change in a local search procedure a new schedule should be counted. Therefore, Lova et al. (2009) define the number of generated schedules as the sum of times each activity of the project has obtained a feasible start time divided by the number of activities of the project. Assume a project with eight activities, each with three modes. Suppose that the SGS generates a schedule based on a activity and mode list (8 start times are assigned) and that a local search procedure has also analyzed the two other (feasible) modes for three of the activities. This means that the procedure has generated and analyzed (8+2x3)/8 = 1,75 schedules. In the remainder of this paper, the last definition is used to define the number of schedules.

5.3.2 Experimental results

5.3.2.1 Results of the PSPLIB and Boctor dataset

The results for the first subset obtained after 5,000 schedules can be found in table 5.2. For the J10 and J20 set, the average deviation from the optimal solution is given. For the J30 and Boctor dataset, the deviation from the minimal critical path-based lower bound is reported. If a solution procedure was not able to obtain a feasible solution for all project instances, the percentage of instances for which a feasible solution was found is indicated between brackets. As can be seen in the table, not all heuristics were able to obtain a feasible solution for all project instances.

The metaheuristics are sorted with respect to decreasing average deviations for the Boctor100 dataset, except for schedules which were not able to obtain feasible results for all datasets. The best results for a specific dataset are displayed in bold. In order to determine the best heuristic, the concept of dominance as defined in Kolisch and Hartmann (2006) is used: a heuristic a is dominated by a heuristic b if a has for at least one instance set a higher deviation than b without having for any of the other combinations a lower average deviation. Table 5.2 reveals that the scatter search and genetic algorithm (as proposed in chapter 3) and the procedure of Lova et al. (2009) are not dominated by other heuristics. For the PSPLIB dataset, our scatter search procedure dominates all other metaheuristics.

As the most recent metaheuristics available in the literature obtain near-optimal results for the PSPLIB dataset, the use of this dataset is not longer recommended to evaluate the performance of new solution procedures. Therefore, the use of the MMLIB50, MMLIB100 and MMLIB+ is encouraged to compare the results of new solution procedures with the existing methods. The results of the existing methods are presented in the next section.

5.3.2.2 MMLIB* and MMLIB+

The results of the computational tests on the MMLIB50, MMLIB100 and MMLIB* are given in the tables 5.3 to 5.5. In these tables, performance measures for the three stop criteria (1,000, 5,000 and 50,000 schedules, respectively) are shown. For every stop criterion, three measures are indicated. First, the percentage of instances for which a feasible solution is found. Second, the average deviation from the minimal critical path based lower bound is given and finally the maximum deviation is shown. The metaheuristics are sorted with respect to the results for 50,000 schedules, except for schedules which were not able to obtain feasible results for all project instances.

Not all metaheuristic solution procedures succeed to obtain a feasible schedule for all project instances. This is mainly due to the fact that these metaheuristics have not included a search procedure to enhance the feasibility of an infeasible solution vector. The procedure of Boctor (1996) and Mori and Tseng (1997) were designed to solve the MRCPSP/R, while the procedures of Bouleimen and Lecocq (2003) and Ranjbar et al. (2009) were also designed to solve other scheduling problems, such as the RCPSP and the DTRTP.

In figures 5.1 to 5.3, a graphic representation is given of three performance measures for the three datasets for each of the three stop criteria and for each metaheuristic. The white part of each bar indicates the percentage of infeasible solutions, while the black part of the bar indicates the percentage of best solutions found. The grey part indicates the feasible solutions for which no best solution could be found. The algorithms are sorted on their percentage of best results found after 50,000 schedules for each of the three datasets.

Based on the obtained results, the following conclusions can be drawn. First,

Author	Repr	SGS	J10	J20	J30	Boctor50	Boctor100
Slowinski et al. (SA)	AL	P	2.01 (44.96)	2.51 (50.18)	6.13 (48.55)	31.70	38.14
Boctor (SA)	AL	S	0.27	11.66 (99.10)	29.58 (93.66)	25.13	26.82
Bouleimen & Lecocq (SA)	AL	S	2.65	10.26 (99.64)	33.60 (99.64)	28.53	31.49
Nonobe and Ibaraki (TS)	AL	P	5.82	9.02	26.43	48.61	56.91
Mori & Tseng (GA)	PL	S	2.68	13.55	24.99	41.28	53.81
Özdamar (GA)	PL	P	0.70	6.05	27.38	37.81	47.21
Jarboui et al. (PS)	PL	S	0.22	2.44	18.14	30.31	37.82
Zhang et al. (PS)	PL	S	0.31	3.17	18.63	28.97	32.83
Van Peteghem & Vanhoucke (AIS)	AL	S	0.02	0.70	14.18	25.83	30.12
Alcaraz et al. (GA)	AL	S	0.24	1.96	21.83	26.48	29.10
Tseng and Chen (GA)	AL	S	0.32	1.47	17.06	26.60	27.12
Hartmann (GA)	AL	S	0.20	1.61	15.96	24.73	26.47
Ranjbar et al. (SS)	AL	S	0.17	1.31	16.21	24.10	25.84
Jozefowska et al. (SA)	AL	S	0.99	6.65	18.64	24.44	25.70
Van Peteghem & Vanhoucke (SS)	PL	S	**0.00**	**0.32**	**13.66**	23.79	25.11
Van Peteghem & Vanhoucke (GA)	PL	S	0.01	0.57	13.75	**23.41**	24.67
Lova et al. (GA)	AL	P/S	0.04	0.89	14.58	23.65	**24.63**

Table 5.2: Average deviation from optimum/critical path lower bound for PSPLIB and Boctor instances after 5,000 schedules

although several solution procedures (Alcaraz et al., 2003; Zhang et al., 2006; Tseng and Chen, 2009) find feasible solutions for all project instances, they do not find any (or a small part) of the best solutions. On the other hand, more than one third of the solutions found by Slowinski et al. (1994) for the MMLIB50 and MMLIB100 dataset is also the best known solution. Second, the infeasibility rate of some procedures clearly decreases for an increasing number of schedules evaluated (Bouleimen and Lecocq, 2003; Mori and Tseng, 1997), while other metaheuristic solution procedures do not succeed to increase this rate significantly (Slowinski et al., 1994; Boctor, 1996; Ranjbar et al., 2009). It is conjectured that this is mainly due to the restricted feasibility improvement methods which are incorporated in the algorithms. Finally, based on these results, it can be stated that our scatter search procedure dominates all other metaheuristics. This is mainly due to the advanced improvement methods which are designed to search for the best performing mode assignment in the search space and to the 'intelligent' solution combination method using resource scarceness characteristics to steer the algorithm to the most promising solution regions. We believe that this objective comparison of various metaheuristic procedures and the creation of new datasets will contribute to the search of new and better solutions in the future.

5.4 Discussion and conclusions

This research has given an overview of the metaheuristic solution procedures available in the literature to solve the multi-mode resource-constrained project scheduling problem and has made a fair comparison between these procedures. Moreover, a new benchmark dataset is proposed. Researchers are encouraged to use this dataset to compare the results of their solution procedures with other procedures.

Based on the results, the following conclusions can be made. First, the multi-mode RCPSP can be divided into two subproblems: the mode assignment problem, whose aim it is to generate a feasible mode assignment list, and a scheduling problem, whose aim it is to minimize the makespan of the problem. Some procedures emphasized too much the scheduling problem, by which these procedures were not able to generate feasible solutions for all project instances. Moreover, procedures using a clever mode assignment procedure to generate the initial population, such as the *minimum normalized resources procedure* of Lova et al. (2009) or the *controlled mode assignment procedure* as presented in chapter 3, have an advantage compared to other methods, especially when the number of generated schedules is low. Future research will have to focus on this mode assignment problem, since for an increasing number of activities and an increasing number of modes the complexity of this problem also increases.

Second, the good performance of some solution procedures is mainly due to the applied MRCPSP-specific local search procedures rather than to the followed metaheuristic search strategy. This conclusion is supported by computational tests executed by several authors comparing the effectiveness of both their algorithm and a similar solution procedure without the proposed local search method. Lova et al. (2009) prove that their hybrid genetic algorithm with efficient improvement method clearly outperforms a simple genetic algorithm. Similar results are found by Hartmann (2001), who shows that the use of the single-pass improvement method clearly improves the performance of the proposed genetic algorithm. Moreover, the use of problem specific information in the local search process, such as the use of resource scarceness parameters in the scatter search procedure, increases the efficiency of the procedure significantly.

Finally, we hope to have contributed in making a fair comparison between the different available datasets. We believe that the introduction of three new datasets opens the possibility to compare new solution procedures with the currently available methods. Although the use of standard datasets often motivates researchers to improve merely the benchmark results rather than to investigate in new promising methodologies, we hope that the introduction of this new dataset will also stimulate the development of new ideas and techniques to tackle the MRCPSP.

Authors	1,000 schedules			5,000 schedules			50,000 schedules		
	Feasible	Av. Dev.	Max. Dev.	Feasible	Av. Dev.	Max. Dev.	Feasible	Av. Dev.	Max. Dev.
Slowinski et al. (SA)	21.85	-	137.50	23.52	-	125.00	25.00	-	121.43
Boctor (SA)	67.59	-	171.43	69.63	-	171.43	78.52	-	213.33
Bouleimen & Lecocq (SA)	72.96	-	234.62	77.78	-	280.00	82.78	-	253.33
Özdamar (GA)	83.15	-	213.33	83.33	-	213.33	83.33	-	206.67
Ranjbar et al. (SS)	85.19	-	206.90	87.04	-	190.00	89.65	-	180.73
Mori & Tseng (GA)	72.22	-	230.77	83.52	-	240.00	100.00	46.91	315.38
Alcaraz et al. (GA)	100.00	56.06	300.00	100.00	43.05	315.38	100.00	40.69	315.38
Nonobe and Ibaraki (TS)	100.00	43.35	284.62	100.00	41.10	276.92	100.00	38.98	261.54
Jarboui et al. (PS)	100.00	49.98	323.08	100.00	38.86	284.62	100.00	32.01	253.85
Zhang et al. (PS)	100.00	49.25	269.23	100.00	35.94	253.85	100.00	30.23	238.46
Tseng and Chen (GA)	100.00	65.17	307.69	100.00	40.99	261.54	100.00	29.44	246.15
Jozefowska et al. (SA)	100.00	49.06	276.92	100.00	33.81	261.54	100.00	27.81	246.15
Hartmann (GA)	100.00	35.40	253.85	100.00	30.61	246.15	100.00	26.81	230.77
Lova et al. (GA)	100.00	34.16	253.85	100.00	28.59	233.08	100.00	26.69	215.38
Van Peteghem & Vanhoucke (AIS)	100.00	31.52	223.08	100.00	27.45	223.08	100.00	25.26	223.08
Van Peteghem & Vanhoucke (GA)	100.00	34.07	246.15	100.00	27.12	215.38	100.00	24.93	215.38
Van Peteghem & Vanhoucke (SS)	100.00	28.17	230.77	100.00	25.45	223.08	100.00	23.79	215.38

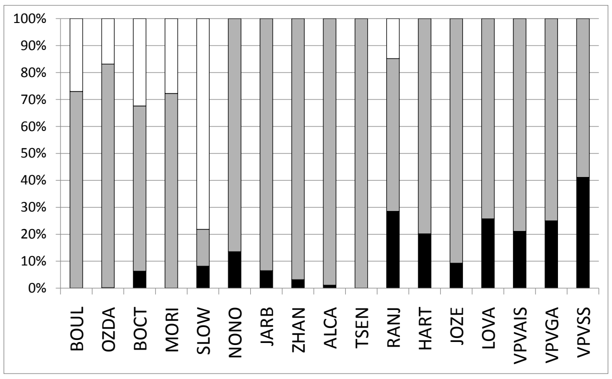
Table 5.3: Average deviation from minimal critical path based lower bound - MMLIB50

Authors	1,000 schedules			5,000 schedules			50,000 schedules		
	Feasible	Av. Dev.	Max. Dev.	Feasible	Av. Dev.	Max. Dev.	Feasible	Av. Dev.	Max. Dev.
Slowinski et al. (SA)	18.89	-	168.18	19.81	-	141.94	21.48	-	140.91
Boctor (SA)	66.67	-	203.57	66.67	-	156.25	66.67	-	150.00
Bouleimen & Lecocq (SA)	66.67	-	200.00	67.04	-	203.57	67.41	-	192.86
Özdamar (GA)	66.67	-	221.43	67.59	-	192.86	67.59	-	192.86
Mori & Tseng (GA)	67.04	-	221.43	69.63	-	247.06	83.33	-	257.14
Ranjbar et al. (SS)	83.33	-	185.71	83.33	-	171.43	83.33	-	169.14
Alcaraz et al. (GA)	100.00	61.80	252.63	100.00	52.67	252.63	100.00	46.68	245.16
Nonobe and Ibaraki (TS)	100.00	49.32	238.10	100.00	47.03	236.84	100.00	45.04	236.84
Jarboui et al. (PS)	91.85	-	300.00	100.00	49.41	247.62	100.00	40.23	231.58
Tseng and Chen (GA)	100.00	72.95	257.89	100.00	66.04	257.89	100.00	37.04	226.32
Zhang et al. (PS)	100.00	57.42	233.33	100.00	44.05	215.79	100.00	35.35	205.26
Jozefowska et al. (SA)	100.00	53.97	258.06	100.00	39.05	238.71	100.00	30.27	205.26
Hartmann (GA)	100.00	39.96	221.05	100.00	33.98	200.00	100.00	29.04	194.74
Van Peteghem & Vanhoucke (AIS)	100.00	36.68	189.47	100.00	31.75	184.21	100.00	28.09	184.21
Lova et al. (GA)	100.00	36.29	216.13	100.00	31.01	200.00	100.00	27.89	194.74
Van Peteghem & Vanhoucke (GA)	100.00	37.58	215.79	100.00	29.55	194.74	100.00	25.63	178.95
Van Peteghem & Vanhoucke (SS)	100.00	29.77	194.74	100.00	26.51	184.21	100.00	24.02	178.95

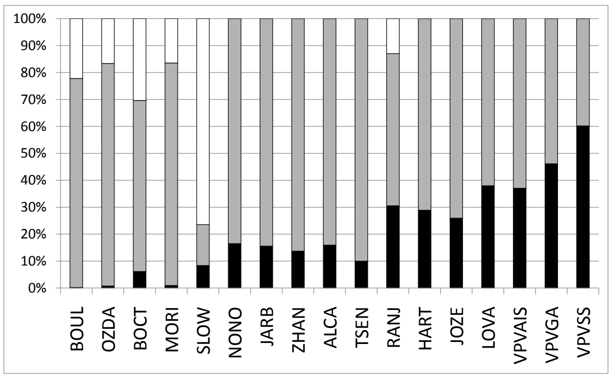
Table 5.4: Average deviation from minimal critical path based lower bound - MMLIB100

Authors	5,000 schedules			50,000 schedules		
	Feasible	Av. Dev.	Max. Dev.	Feasible	Av. Dev.	Max. Dev.
Slowinski et al. (SA)	4.78	-	307.32	4.78	-	307.32
Boctor (SA)	54.48	-	427.78	65.77	-	490.63
Özdamar (GA)	68.64	-	528.13	68.77	-	500.00
Bouleimen & Lecocq (SA)	67.96	-	606.25	70.59	-	590.63
Ranjbar et al. (SS)	70.66	-	805.05	73.65	-	623.15
Jozefowska et al. (SA)	75.00	-	810.34	75.00	-	721.88
Mori & Tseng (GA)	72.59	-	702.70	97.22	-	1057.69
Van Peteghem & Vanhoucke (GA)	97.59	-	793.10	97.59	-	775.86
Zhang et al. (PS)	99.81	-	865.52	99.85	-	848.28
Alcaraz et al. (GA)	100.00	177.55	1000.00	100.00	164.87	996.55
Tseng and Chen (GA)	100.00	183.02	996.55	100.00	142.14	917.24
Nonobe and Ibaraki (TS)	100.00	148.02	920.69	100.00	143.66	910.34
Jarboui et al. (PS)	94.29	-	993.10	100.00	137.99	913.79
Hartmann (GA)	100.00	132.01	872.41	100.00	111.45	793.10
Lova et al. (GA)	100.00	114.07	748.28	100.00	102.73	700.00
Van Peteghem & Vanhoucke (AIS)	100.00	112.82	775.17	100.00	101.95	700.00
Van Peteghem & Vanhoucke (SS)	100.00	101.45	734.48	100.00	92.76	679.31

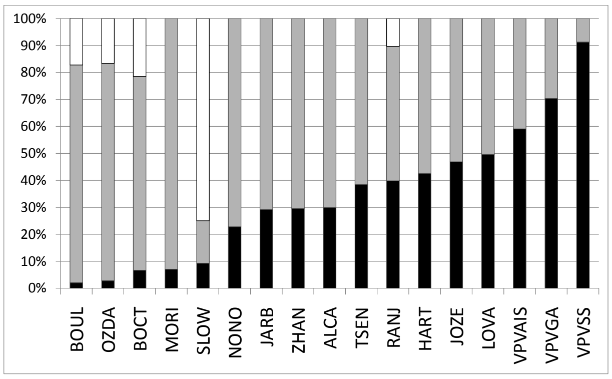
Table 5.5: Average deviation from minimal critical path based lower bound - MMLIB+



(a) 1,000 schedules

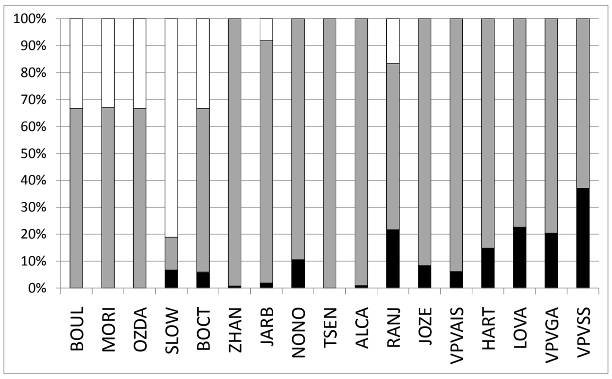


(b) 5,000 schedules

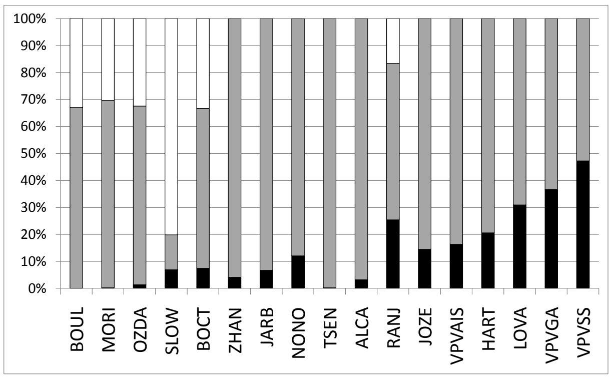


(c) 50,000 schedules

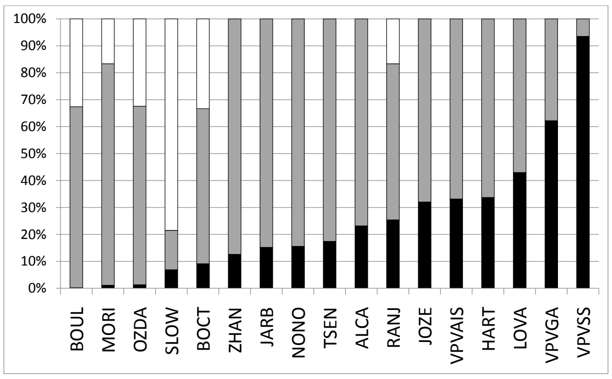
Figure 5.1: Computational performance on MMLIB50



(a) 1,000 schedules

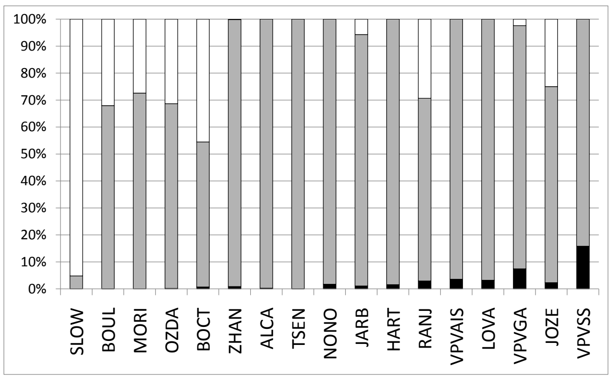


(b) 5,000 schedules

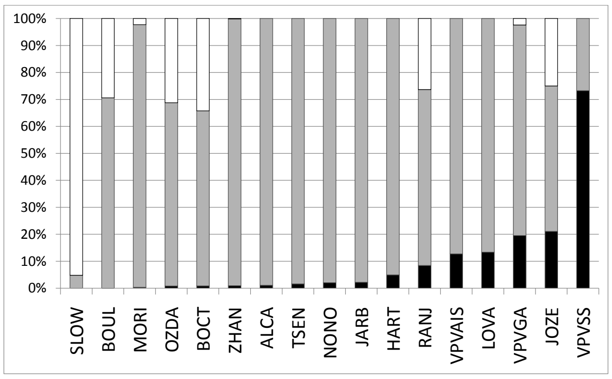


(c) 50,000 schedules

Figure 5.2: Computational performance on MMLIB100



(a) 5,000 schedules



(b) 50,000 schedules

Figure 5.3: Computational performance on MMLIB+

Part II

Extensions to the MRCPSP

6 Preemption

6.1 Introduction

The basic RCPSP and MRCPSP assume that each activity, once started, will be executed until its completion. The extension to the preemptive multi-mode version (P-MRCPSP) allows activities to be preempted at any integer time instance and restarted later on at no additional cost. For the single-mode case, Kaplan (1988), Demeulemeester and Herroelen (1996) and Vanhoucke and Debels (2008) present exact algorithms, while Damay et al. (2007) propose a linear programming based algorithm. Ballestin et al. (2008) reveal the benefits of allowing only 1 interruption per activity and prove the effectiveness of justification in the presence of preemption. Ballestin et al. (2009) investigate the effect of interruption on the project makespan in more general cases and analyze the usefulness of preemption in the presence of due dates. For the multi-mode case, Buddhakulsomsiri and Kim (2006) prove that preemption is very effective to improve the optimal project makespan in the presence of resource vacations and temporary resource unavailability and that the makespan improvement is dependent on the parameters that impact resource utilization. To the best of our knowledge, no further research has been performed on the P-MRCPSP.

In this chapter, a genetic algorithm approach for solving the P-MRCPSP is introduced. In section 6.2 the problem formulation of the P-MRCPSP is presented, while in section 6.3 the details of the solution procedure are described. In section 6.4, the computational results for the MRCPSP with activity preemption executed on the PSPLIB and Boctor dataset are presented. Finally, the conclusions of this chapter are summarized in section 6.5.

6.2 Problem formulation

The P-MRCPSP can be stated as follows. The project is represented as an activity-on-the-node network $G(N, A)$, where N is the set of activities and A is the set of pairs of activities between which a finish-start precedence relationship with a minimal time lag of 0 exists. A set of activities, numbered from 1 to $|N|$ with a dummy start node 0 and a dummy end node $|N| + 1$, is to be scheduled on a set R^ρ of renewable and R^ν of nonrenewable resource types. Each activity $i \in N$ is performed in a mode m_i, which is chosen out of a set of $|M_i|$ different execution modes $M_i = \{1, ..., |M_i|\}$. The duration of activity i, when executed in mode m_i, is d_{im_i}. Moreover, since activities are allowed to be preempted at any integer time and restarted later on at no additional cost, the model employs one-period subactivities (see Kolisch et al., 1995). Therefore, each duration unit v of an activity i scheduled in mode m_i (with $v \in \{0, ..., d_{im_i} - 1\}$) is assigned a start time s_{iv}. Each mode m_i requires $r^\rho_{im_ik}$ renewable resource units ($k \in R^\rho$). For each renewable resource $k \in R^\rho$, the availability a^ρ_k is constant throughout the project horizon. Activity i, executed in mode m_i, will also use $r^\nu_{im_il}$ nonrenewable resource units ($l \in R^\nu$) of the total available nonrenewable resource a^ν_l. A schedule S is defined by a vector of activity start times s_{iv} and a vector denoting its corresponding execution modes m_i. A schedule is said to be feasible if all precedence and renewable and nonrenewable resource constraints are satisfied. The objective of the P-MRCPSP is to minimize the makespan of the project.

The preemptive problem can be formulated as follows:

$$\text{Min. } s_{n+1,0} \tag{6.1}$$

s.t.

$$s_{i,d_{im_i}-1} + 1 \leq s_{j,0} \qquad \forall (i,j) \in A \tag{6.2}$$

$$s_{i,v-1} + 1 \leq s_{i,v} \qquad \forall i \in N, \forall v \in \{1, d_{im_i} - 1\} \tag{6.3}$$

$$\sum_{i \in S(t)} r^\rho_{im_ik} \leq a^\rho_k \qquad \forall k \in R^\rho, \forall m_i \in M_i \tag{6.4}$$

$$\sum_{i=1}^{|N|} r^\nu_{im_il} \leq a^\nu_l \qquad \forall l \in R^\nu, \forall m_i \in M_i \tag{6.5}$$

$$m_i \in M_i \qquad \forall i \in N \tag{6.6}$$

$$s_{0,0} = 0 \tag{6.7}$$

$$s_{i,v} \in \text{int}^+ \qquad \forall i \in N, \forall v \in \{0, d_{im_i} - 1\} \tag{6.8}$$

where $S(t)$ denotes the set of activities in progress in period $[t-1, t[$, $t \in \{1, ..., s_{n+1}\}$. The objective function 6.1 minimizes the total makespan of the project. In constraint set 6.2, the earliest start time of an activity j cannot be smaller than the finish time for the last unit of duration of its predecessor i. Constraint set 6.3 guarantees that the start time for every time instance of an activity has to be at least 1 time unit larger than the start time for the previous unit of duration. Constraints 6.4 and 6.5 take care of the renewable and nonrenewable resource limitations, respectively. Each activity i has to be performed in exactly one mode m_i (constraint 6.6). Constraint 6.7 forces the project to start at time instance 0 and constraint 6.8 ensures that the activity start times assume nonnegative integer values. A schedule which fulfills all the constraints 6.1 to 6.8, is called *optimal*.

Resume the example project given in chapter 2. If we relax this example project to the P-MRCPSP, a schedule as shown in figure 6.1 can be generated. This schedule is based on the mode list (1,1,2,1,2,1,2,2) and is scheduled according to the same RK as schedule 2.2(b). The activities 1 and 5 are preempted once (the different parts are indicated as $1_1\&1_2$ and $5_1\&5_2$, respectively), as can be seen in the figure. In the next section, the optimal solution for this problem is shown.

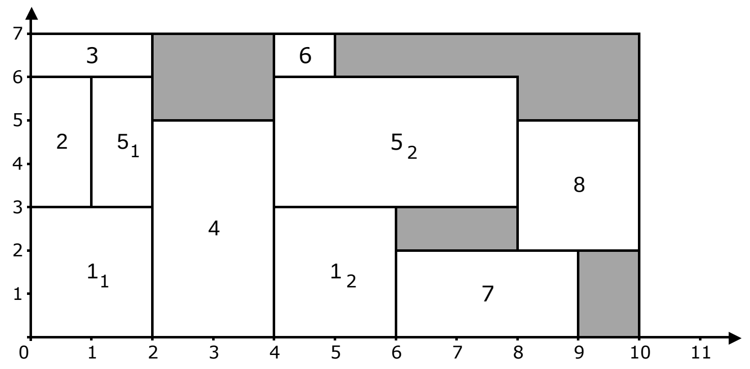


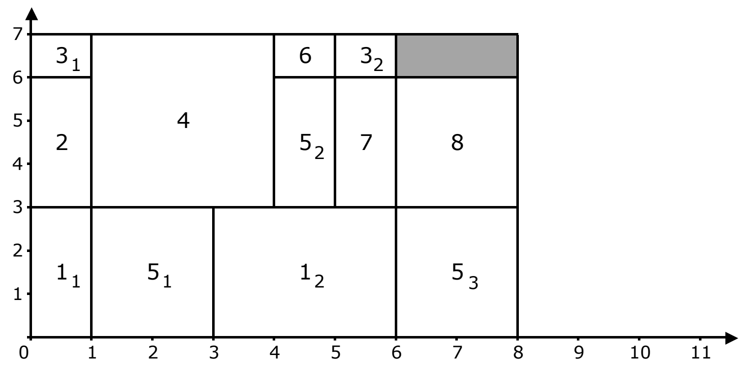
Figure 6.1: Feasible schedule P-MRCPSP

6.3 Solution procedure

The bi-population genetic algorithm, as proposed in section 3.3 of chapter 3, is used as the framework for the solution procedure for the P-MRCPSP. In order to allow activity preemption, the original activity network is converted to a new network. From the moment a mode m_i is assigned to an activity i using the mode list μ, its duration d_{im_i} is known. Afterwards, each activity is split into d_i subactivities with a unit duration of 1, resulting in the new constructed network (see

Demeulemeester and Herroelen, 1996; Vanhoucke and Maenhout, 2009). The vector λ, which now determines a schedule sequence for all subactivities of the new project network, determines the subactivity start times and hence the algorithm can now be used with no further changes. The mode improvement procedure is only applied to activities that have been scheduled completely (i.e. for which all subactivities are scheduled).

The optimal solution for the preempted problem is shown in figure 6.2.



(a) P-MRCPSP

Figure 6.2: Optimal solution for the example project for the P-MRCPSP

6.4 Computational results

This section presents computational results for the MRCPSP with activity preemption. Table 6.1 summarizes the results obtained from tests on the PSPLIB data instances J10, J20 and J30 with and without the presence of the nonrenewable resources. Results are also shown for the Boctor50 and Boctor100 data instances (Boctor, 1993), which contain projects without nonrenewable resources. The second and third columns display the average deviation from the critical path based lower bound for the MRCPSP and P-MRCPSP, respectively. The column with label 'Av.Impr.' displays the average makespan improvement for the P-MRCPSP relative to the MRCPSP. The columns with labels 'Better', 'Equal' and 'Worse' show the number of preemptive solutions with a lower, equal or higher project makespan than the solutions found for the MRCPSP. The results in table 6.1 can be used for comparison purposes for future research in this area. The computational results for the different datasets can be downloaded from www.projectmanagement.ugent.be.

Table 6.1 shows that activity preemption obviously leads to an overall average makespan improvement. However, the PSPLIB instances show that a frac-

Table 6.1: Results for different datasets with and without preemption - 5,000 schedules

		MRCPSP	P-MRCPSP	Av.Impr.	Better	Equal	Worse
NR	J10	32.27%	31.54%	0.49%	47	488	1
	J20	17.76%	17.03%	0.55%	99	432	23
	J30	13.75%	13.23%	0.41%	108	404	40
No NR	Boctor50	23.41%	21.52%	1.55%	111	9	0
	Boctor100	24.67%	21.91%	2.22%	120	0	0
	J10*	15.49%	14.94%	0.42%	36	500	0
	J20*	8.27.%	7.61%	0.53%	68	486	0
	J30*	5.60%	5.06%	0.44%	74	478	0

* = ignoring the nonrenewable resources (NR)

tion of the projects shows a higher makespan with preemption compared to the best found MRCPSP solution. Indeed, introducing activity preemption results in a larger project network, since each non-preempted activity needs to be splitted into subactivities, leading to an increase of the random key size. This larger RK search space is responsible for the fraction of solutions displayed in the 'Worse' column.

The table also shows that activity preemption always leads to better solutions (the column 'Worse' is equal to 0) when no nonrenewable resources are taken into account. In that case, infeasible mode assignments will never occur, leaving more room to the algorithm to select low duration modes. Obviously, when nonrenewable resources are taken into account, the low durations correspond to relatively high nonrenewable resource demand, leading to infeasible mode assignments. We indeed observe that the best found solutions without nonrenewable resources contain many mode assignments with low activity durations, which is in line with the study of Boctor (1993) who has shown that the 'shortest feasible mode' selection rule performs best.

The results of the introduction of preemption on the new datasets MMLIB50 and MMLIB100 are presented in table 6.2. The results are in line with the results obtained for the PSPLIB and Boctor dataset: in case nonrenewable resources are taken into account, the procedure is not always able to find similar or better results than in the non-preempted case. In case no nonrenewable resources are considered, no inferior results are obtained.

6.5 Conclusions

In this chapter, we have designed an adapted version of the genetic algorithm for the preemptive multi-mode resource-constrained project scheduling problem. The results of the computational tests performed on the PSPLIB dataset revealed that the introduction of preemption does significantly help in decreasing the average project makespan compared to the non-preempted case. If no nonrenewable resources are taken into account, the introduction of preemption always leads to better solutions.

Table 6.2: Results for the MMLIB dataset with and without preemption - 5,000 schedules

		MRCPSP	P-MRCPSP	Av.Impr.	Better	Equal	Worse
NR	MMLIB50	20,164	20,092	0.30%	154	299	87
	MMLIB100	25,623	25,557	0.16%	169	249	122
No NR	MMLIB50*	17,149	17,032	0.62%	94	446	0
	MMLIB100*	21,839	21,583	0.98%	133	407	0

* = ignoring the nonrenewable resources (NR)

7 Introduction of Learning Effects

7.1 Introduction

Project scheduling has been a research topic for many decades, resulting in a wide variety of optimization procedures. The main focus on the project lead time minimization has led to the development of various exact and (meta)heuristic procedures for scheduling projects with tight resource constraints under a wide variety of assumptions. The basic problem type in project scheduling is the well-known resource-constrained project scheduling problem (RCPSP). This problem type aims at minimizing the total duration or makespan of a project subject to precedence relations between the activities and the limited renewable resource availabilities, and is known to be NP-hard (Blazewicz et al., 1983).

In most projects, human resources are a critical factor in the scheduling process. Not only their availability, but also their productivity will influence the project duration. One of the reasons why the productivity of a human resource varies over time is the effect of learning (Wright, 1936), which indicates the process of acquiring experience while performing similar activities leading to an improvement of the worker's skill. As a measurable result of learning, the time required to perform the next jobs decreases (Janiak and Rudek, 2007).

However, in the project scheduling literature, most models assume static and often homogeneous efficiencies of resources (Heimerl and Kolisch, 2009). In this chapter, the effect of learning on the efficiency of human resources is studied for the well-known discrete time/resource trade-off problem (DTRTP). In this scheduling problem, each activity contains a specific work content in terms of working days, instead of a fixed duration and resource requirement. For each activity, a set of execution modes can be specified using different combinations of durations and resource requirements, as long as the specified work content is met. The objective of this problem type is to minimize the total duration or makespan of the project.

In this chapter, we analyze the influence of the introduction of learning effects in project scheduling and determine the driving variables which can explain the difference between a schedule with and without learning effects. Moreover, the influence of learning effects on the accuracy of resource-constrained project schedules is investigated, in order to provide insights in the scheduling process with learning effects and to supply managerial understandings to optimize the decision process. The relevance and contribution of this research study is given by Bochenski (1993) who states that project managers can obtain a competitive advantage by incorporating learning effects in order to obtain better deadlines and to use the available resources more efficiently.

The remainder of this chapter is organized as follows. In the remainder of this section, a literature overview and the modeling aspects of learning effects applied to scheduling problems is given. In section 7.2, the discrete time/resource trade-off problem is presented and the mathematical modeling of learning effects in this scheduling problem is given. Section 7.3 discusses the solution approach while section 7.4 gives an overview of the purpose and design of the computational experiment. The results for the computational experiments are shown in section 7.5, while in section 7.6, the conclusions of this research are formulated.

7.1.1 Literature overview

Wright (1936) was the first to describe the link between working costs per unit and the production output in the aircraft industry. He discovered that for every redoubling of the output, the unit processing time of an aircraft decreases by 20%. This empirical phenomenon, where the cumulative average worker hours will decline by a certain percentage of the previous cumulative average rate when the production quantity of a product doubles, was observed in various scientific areas since then. For an overview of the available learning models the reader is referred to Nembhard and Uzumeri (2000).

For decades, researchers were convinced that learning effects were the result of the repetitive execution of activities or tasks. However, more recent literature distinguishes two different groups of learning:

- *Autonomous learning* results from repeating similar operations, leading to a higher familiarization and routine (Biskup, 2008). The productivity of the workforce increases due to experience (earlier performance, mistakes, ...) and these experiences grow by repeatedly executing specific activities or tasks. This type of learning is also referred to as *learning-by-doing* or the *Horndal-effect*.

- *Induced learning* is the result of management investing in the know-how and the productivity of the human resources (Adler and Clark, 1991; Upton and Kim, 1998). Additional training, changes in the remuneration system and production environment, innovative education or incentive schemes are examples of management decisions that can influence the learning rate of the human resources. Important in this case is the determination of the optimal learning rate, since a reduction of the learning rate is usually related to higher costs.

Learning effects have been discussed in the scheduling literature from various angles.

In a review paper on the different learning models proposed in the *machine scheduling* literature, Biskup (2008) makes a distinction between *position-based learning*, where the learning is only affected by the number of tasks being processed, and the *sum-of-processing-time learning*, that takes into account the processing time of all jobs processed so far.

In the *staff scheduling* literature, several authors already modeled efficiency to cope with the effects of learning. Wu and Sun (2006) denote that the efficiency of staff will improve by doing more. They develop a genetic algorithm to minimize the total outsourcing costs for a project with a fixed time horizon. Heimerl and Kolisch (2009) present an optimization model to address the problem of assigning project work to multi-skilled internal and external human resources while considering learning, depreciation of knowledge and company skill level targets.

In *multi-project scheduling*, Shtub et al. (1996) and Amor and Teplitz (1998) develop heuristic and metaheuristic procedures for scheduling projects with a repetitive nature under learning effects. Ash and Smith-Daniels (1999) present a heuristic for a multi-project scheduling problem where project preemption is allowed and learning, forgetting and relearning effects are introduced. They state that, during the execution of an activity in development projects, people continuously learn and become more activity-efficient. They define learning as the increased productivity during the course of each project activity (*activity-specific learning*).

The empirical evidence that learning takes place over time can be found in Brazel (1972) and Sahal (1979), who show that the same power function learning model as the repetition-based learning model noted by Wright (1936) can be used. Hanakawa et al. (1998) reveal that learning effects exist in project settings in general and software development settings in particular. They show that the productivity of a developer will increase if he/she stays on the task longer. The variations of the productivity depend on the developer's knowledge, experience and learning curves. Another behavioral study of Hendriks et al. (1999) reveals that two factors can affect the staff allocation: the *project scatter*, which indicates that the team efficiency will increase if the number of team staff is larger than needed, and the *resource dedication*, which denotes that the dedication of a staff member to a particular job can increase efficiency.

The focus of this work is on the introduction of *autonomous* and *activity-specific* learning curves in a multi-mode project scheduling environment.

7.1.2 Modeling

The introduction of the learning concepts in scheduling environments requires a mathematical formulation. The following assumptions and definitions are formulated:

- The *efficiency* of a human resource is defined as the portion of work performed in one time unit by a human resource, assuming that the entire task takes one time unit for an employee working at a normal level (i.e. efficiency = 100%) (Gutjahr et al., 2008).
- A person working at a uniform efficiency level of 100% during one day, is performing one *working day*.
- A *man-day* is defined as the period of time needed to execute the work performed on one working day by one person, taking into account its average efficiency level.
- The work content of a specific activity is expressed in working days. In project scheduling environments without learning effects, the number of working days to execute an activity is equal to the number of man-days. When learning effects are considered, the number of man-days to execute the work content of an activity depends on the activity duration and the values of the learning variables (i.e. efficiency).
- Throughout this chapter, it is assumed that every resource performing activity *i* has the same initial average efficiency level $\overline{E}_{i1}$ and learning rate L_i (denoted by $\overline{E}_1$ and L). This is in contrast with other authors (see e.g. Gutjahr et al., 2008) who assume that every employee possesses different knowledge, education, skills, abilities, etc. in different fields.

The traditional learning curve is a downward sloping power function that represents the amount of time spent to complete the next iteration of a single task and is frequently formalized by using the following formula:

$$T_n = T_1 n^{-b} \tag{7.1}$$

where T_n is the average processing time of the n-th unit of the cumulative production quantity n, T_1 the production time of the first unit and $b = -log_2 L$ the learning index depending on the learning rate L, which indicates the learning achieved. Thus, the lower the learning rate L, the higher the effects from learning will be.

Average efficiency Based on this traditional learning curve, Wu and Sun (2006) defined the *average efficiency curve*, representing the activity-specific knowledge of a resource after a time period t for a specific activity i, as follows:

$$\overline{E}_{it} = \overline{E}_{i1} t^{b_i} \tag{7.2}$$

where $\overline{E}_{i1}$ indicates the initial efficiency curve of activity i and b_i is the learning index of activity i depending on the activity-specific learning rate L_i. This formula indicates that when a human resource works longer on an activity, his or her average activity-specific efficiency level and competency increases. In figure 7.1(a), the average efficiency curve is shown for a human resource with an initial average efficiency $\overline{E}_1$ of 0.7 and a learning rate L of 0.85. Based on the formulation, one can calculate that the average efficiency at time 10 is $0.7 * 10^{-log_2 0.85} = 1.201$.

Real efficiency The average efficiency curve only indicates the average of the (actual) efficiency values over the time period [0,t]. As shown, the average efficiency at time 10 in the example is 1.201 and indicates the average of all the efficiencies over the period [0,10]. This leads to a total of 12.01 working days executed during the first 10 days (1.201 * 1 day) (see also figure 7.1(a)). The same result is obtained by calculating the surface under the *real efficiency curve*, which indicates the efficiency E_{it} of a resource i on a specific time instant t. The real efficiency E_{it} can be formulated as follows:

$$E_{it} = \overline{E}_{i1}(1 + b_i)t^{b_i} \tag{7.3}$$

The efficiency curve is shown in figure 7.1(b). The efficiency at time 10 is 1.483, which is obviously higher than the average efficiency at that time instant. The surface under the efficiency curve for the period [0,10] is 12.01 which is obviously equal to the surface of the rectangle in figure 7.1(a).

Number of man-days The formula of the efficiency curve will be used to calculate the number of man-days needed to execute a specific work content. Therefore, the *number of working days performed by one human resource during T man-days* (D_T^w) is first calculated. Since the assumption is made that the work content is expressed in working days, D_T^w can be calculated as:

$$D_T^w = \int_{t=0}^{T} \overline{E}_{i1}(1+b_i)t^{b_i} = \frac{\overline{E}_{i1}(1+b_i)}{(1+b_i)}T^{1+b_i} = \overline{E}_{i1}T^{1+b_i} \qquad (7.4)$$

In figure 7.1(c) the D_T^w-curve is shown which indicates the number of working days executed during t man-days. As can be seen, 12.01 working days are executed after 10 man-days. The curve $x = y$ bisects the area and reveals that after 4.6 man-days the number of working days executed is larger than the number of man-days and this is due to the increasing efficiency level of the human resource.

To calculate the *number of man-days needed to execute T working days* (D_T^m), the formula of D_T^w is reversed, which gives the following formula:

$$D_T^m = \sqrt[1+b_i]{\frac{T}{\overline{E}_{i1}}} \qquad (7.5)$$

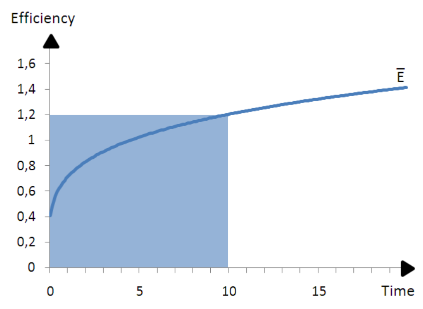
In figure 7.1(d) the D_T^m-curve is shown which indicates the number of man-days needed to execute t working days. For example, to execute 10 working days 8.62 man-days are needed.

7.2 DTRTP

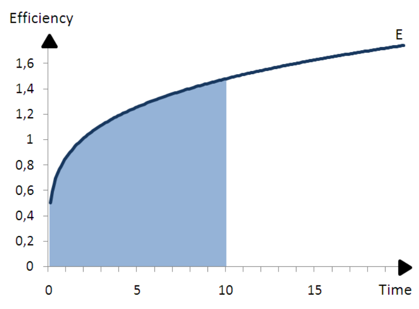
Based on the definitions of D_T^m and D_T^w, the concept of learning can now be introduced in the discrete time/resource trade-off problem (DTRTP). In section 7.2.1, the mathematical formulation of the DTRTP is explained and a literature overview is presented. In section 7.2.2, the learning concept is introduced in the DTRTP and an example is given.

7.2.1 Problem formulation

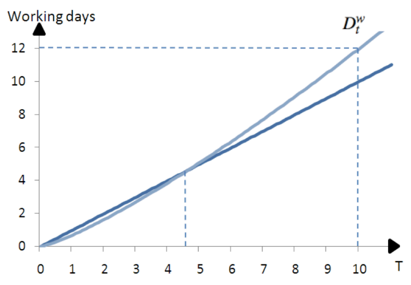
The discrete time/resource trade-off problem (DTRTP) is a subproblem of the multi-mode resource-constrained project scheduling problem (MRCPSP) and can be formulated as follows. An activity-on-the-node network $G(N, A)$ is presented, consisting of a set of nodes N and a set of activities A, representing the precedence relations, between which a finish-start precedence relationship with a minimal time lag of 0 exists. A single renewable resource with constant availability a is available throughout the project horizon. The set N contains a set of $|N|$ activities, numbered from 1 to $|N|$ with a dummy start node 0 and a dummy end node $|N| + 1$, representing the start and completion of the project, respectively. Each activity $i \in N$ contains a specific work content w_i, e.g. expressed in working days. For each activity, a set of allowable execution modes $|M_i|$ could be specified. The activity i, when executed in an efficient mode m_i, has a duration of d_{im_i} and a resource demand of r_{im_i} renewable resource units, such that $d_{im_i} r_{im_i}$ is at least



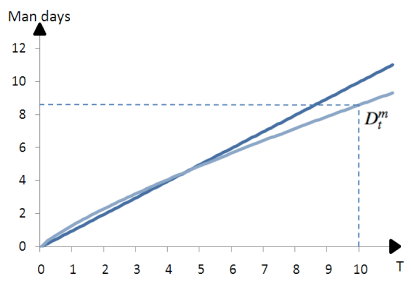
(a) Average efficiency curve



(b) Efficiency curve



(c) D_T^w-curve



(d) D_T^m-curve

Figure 7.1: Mathematical modeling of average and real efficiency curves and the number of working days and man-days with and without learning

equal to and as close as possible to w_i. A mode is called efficient if every other mode has either a strictly higher duration or a strictly higher resource demand. The objective is to schedule each activity in one of its execution modes, subject to both the precedence and resource constraint, under the objective to minimize the project makespan. The problem can be formulated as follows:

$$\begin{aligned}
&\text{minimize } f_n && && (7.6)\\
&\text{subject to } f_i + d_{jm_j} \leq f_j && \forall (i,j) \in A && (7.7)\\
&\sum_{i \in S(t)} r_{im_i} \leq a && \forall m_i \in M_i, \forall t && (7.8)\\
&m_i \in M_i && \forall i \in N && (7.9)\\
&f_0 = 0 && && (7.10)\\
&f_i \in \text{int}^+ && \forall i \in N && (7.11)
\end{aligned}$$

where $S(t)$ denotes the set of activities in progress in period $]t-1, t]$ and f_i the finish time of the i^{th} activity.

Demeulemeester et al. (2000) present an efficient branch-and-bound approach to solve the DTRTP, while De Reyck et al. (1998) present several heuristic procedures, based on local search procedures and Tabu Search (TS). Recently, Ranjbar and Kianfar (2007) have developed a genetic algorithm and Ranjbar et al. (2009) presented a scatter search procedure. In this work, we will use an adapted version of the bi-genetic algorithm of Van Peteghem and Vanhoucke (2010), who present state-of-the-art results for the multi-mode resource-constrained project scheduling problem. The adaptions will be discussed in detail in section 7.3.1.

To incorporate the learning concept into this model, equation 7.11 should be removed and equation 7.7 should be defined as

$$f_i + \sqrt[1+b_j]{\frac{d_{jm_j}}{\overline{E}_{j1}}} \leq f_j \qquad \forall (i,j) \in A \qquad (7.12)$$

with b_j the learning rate and $\overline{E}_{j1}$ the initial efficiency. In case no learning effects occur (i.e. L_j=100% and $\overline{E}_{j1}$=100%), the value $\sqrt[1+b_j]{\frac{d_{jm_j}}{\overline{E}_{j1}}}$ is equal to d_{jm_j}.

7.2.2 Learning effects in the DTRTP

For every activity in the DTRTP, the work content, expressed in working days, is known in advance and based on this work content, several execution modes can be determined. Consider an activity i with a work content of 12 working days.

Ten human resources are available and the following five modes (d_{im_i}, r_{im_i}) are determined: (2,6), (3,4), (4,3), (6,2) and (12,1).

Consider mode 2, which needs 4 resources, each performing 3 working days. In case homogeneous efficiencies are assumed, the learning variables L and $\overline{E}_1$ are equal to 100%, which means that the efficiency curve remains fixed at an efficiency level of 100% (curve E_H in figure 7.2). Obviously, in this case the number of working days equals the number of man-days.

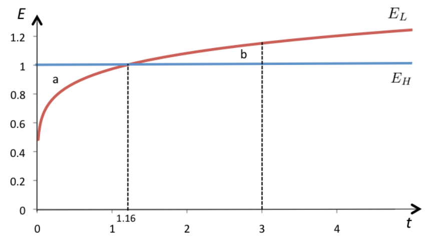


Figure 7.2: Efficiency curve of example project

However, in the presence of learning effects, the efficiency curve will be influenced by the learning variables L and $\overline{E}_1$. Suppose a learning rate of $L = 0.90$ (b = 0.152). To obtain a number of working days which is equal to the number of man-days in mode 2, the initial efficiency should be equal to 0.846 ($3/3^{1.152}$ - see equation 7.3). This results in an efficiency curve E_L for which the surface under the curve is equal to the surface under the homogeneous efficiency curve E_H. In other words, the areas *a* and *b* in figure 7.2 are equal.

Due to the learning effects, the efficiency curve is not longer homogeneous. From time unit 0 to 1.16, the resources have an efficiency which is lower than 100%, however, after time unit 1.16 the efficiency exceeds the 100%. After 3 man-days, a total of 3 working days is performed by each resource, which results in a total work content (for the 4 resources) of 12 working days.

However, when another mode will be selected and the same learning variables are used, the introduction of learning can lead to positive or negative effects on the total activity duration. Assume the same learning variables for mode 1, in which 6 resources have to perform 2 working days. After two man-days these resources have only executed 6 x 1.88 working days (surface under the efficiency curve from 0 to 2 = D_2^w = 0.846 x $2^{1.152}$) = 11.28 working days. However, to execute in total 12 working days, these 6 resources each have to perform D_2^m = $\sqrt[1.152]{\frac{2}{0.846}}$ = 2.11 man-days, which is 0.11 days longer than in the homogeneous case. Consequently, learning has a negative effect when mode 2 will be chosen. If under the same conditions mode 3 is chosen, these 3 resources only have to execute 3.85 man-days to execute 12 working days since efficiency improvements are obtained for a longer period. In this case, introducing learning had a positive effect on the duration of mode 3. An overview of the other execution modes can be found in table 7.1.

mode	d_{im}		r_{im}	w_i	
	homogeneous	*learning*		*homogeneous*	*learning*
1	2	2.11	6	12	12.66
2	3	3	4	12	12
3	4	3.85	3	12	11.55
4	6	5.48	2	12	10.96
5	12	9.99	1	12	9.99

Table 7.1: Duration and total work content with and without learning

As can be seen in the table, the introduction of learning effects introduces a new trade-off between the activity duration and the total work content. Since working longer on an activity will decrease the total work content expressed in man-days, it becomes more interesting to execute the activity in a mode with a smaller resource demand since these resources will become more efficient when they work longer on the activity. However, shorter execution modes are in favor of the objective to minimize the total makespan of the project, although their total work content will be larger.

7.3 Solution approach

This section briefly discusses the adaptations made on an existing solution procedure in order to take the learning effects into account, and shows that this procedure can compete with current state-of-the-art solution procedures for the DTRTP without learning effects.

7.3.1 Solution procedure

In order to test the impact of activity learning on the total project duration, an adapted version of the bi-population based genetic algorithm of Van Peteghem and Vanhoucke (2010) has been used. This procedure has been originally developed to solve multi-mode resource-constrained project scheduling problems within the presence of renewable and nonrenewable resource constraints. In order to incorporate learning curves during the construction of a baseline schedule, the following adaptions have been made:

- Since the introduction of the learning effect creates non-integer durations for each of the different modes, the schedule generation scheme has been adapted to deal with these fractional durations.

- In order to keep the populations diverse, each new population element that is not diverse in terms of mode assignment and sequence of the activities, is changed randomly until the element is diverse enough with respect to the other elements in the population.
- The 10% population elements with the highest project makespan are replaced with new randomly generated population elements. To determine the mode list of the new population elements, an inverse frequency function is used to stimulate the use of infrequently used execution modes.

The computational experiment of the next section is set up to show that our solution procedure can compete with state-of-the-art solution procedures to solve the DTRTP without activity learning available in the literature. In section 7.4, our solution procedure is then used to analyze the effect of learning on the quality of the project schedule through a detailed computational experiment.

7.3.2 Results

In order to compare our algorithm with state-of-the-art DTRTP algorithm without learning from the literature, we rely on the dataset proposed by De Reyck et al. (1998) and used in several papers. The set contains project instances with 10, 15, 20, 25 or 30 activities and an order strength (Mastor, 1970) equal to 0.25, 0.50 or 0.75. The work content of every activity lies between 10 and 100. For every combination 10 projects have been generated, leading to a total of 5 x 3 x 10 = 150 different project instances. These problem instances were tested for several resource availabilities (from 10 to 50 in steps of 10) and for several numbers of modes (from 1 to 6, and a version with an unlimited number of modes). In total 150 x 5 x 7 = 5,250 problem instances were tested.

The modes are generated in line with the procedure of De Reyck et al. (1998) as follows: The procedure generates the first mode ($m_i = 1$) for activity i with duration d_{i1} and resource requirement $r_{i1} = \lceil w_i/d_{i1} \rceil$ with $d_{i1} = \lceil w_i/a \rceil$ when the number of modes is restricted and $d_{i1} = max(\lfloor \sqrt{w_i} \rfloor, \lceil w_i/a \rceil)$ when it is unrestricted. Then the procedure generates the second mode with duration $d_{i1} + 1$ and corresponding resource requirements. This new mode is accepted as the second mode if at least one of the resource requirements is different. This mode generation process continues until the desired number of modes is reached or no more modes are left.

Although the focus of this work is on the effects of the introduction of learning effects in project scheduling and not on the development of a new state-of-the-art procedure to solve the DTRTP, promising results are obtained compared to the best results available in the literature, as can be seen in table 7.2. The average deviation from the lower bound (Av. Dev. LB) and the maximum deviation from the lower bound (Max. Dev. LB) are shown after 1 second. The lower bound is calculated as the maximum of the critical path-based lower bound and the resource-based lower bound. The tests were performed on a PC Pentium IV, 3GHz processor with 1GB internal memory, which is comparable to the one used by Ranjbar et al. (2009).

	Av. Dev. LB	Max. Dev. LB
De Reyck et al. (B&B)	2.73	43.75
Ranjbar et al. (SS)	2.77	42.86
This work (GA)	2.88	42.86
Ranjbar and Kianfar (GA)	3.06	42.86
De Reyck et al. (TS)	3.26	91.67

Table 7.2: Comparative results for DTRTP without learning effects (1 sec)

7.4 Experimental design

In this section, the settings of a computational experiment that compare and validate the impact of learning on the project schedule are described. To that purpose, three different project schedules are defined in order to make three different comparisons. The definition of the schedules and their link with the research design are discussed in sections 7.4.1 and 7.4.2, respectively. Section 7.4.3 gives an overview of the data used throughout the experiment. In section 7.5, the results of these experiments are discussed in detail.

7.4.1 Schedule generation

This section presents three different project schedules that will be used throughout the remainder of the computational experiment. Each schedule has a specific purpose and is constructed under different assumptions. The construction and interpretation of the three schedules will be explained here by the use of a fictitious example project, and their use will be explained in section 7.4.2. The example project contains 8 non-dummy activities, with each activity i having a work content w_i above each node of the activity-on-the-node network of figure 7.3(a). The availability of the single renewable resource is equal to 10. The value of L is set at 0.90 and $\overline{E}_1$ is equal to 0.7. Table 7.4 gives for each activity an overview of the possible execution modes (restricted to maximum 6 modes per activity), the best mode for each of the three schedules S^O, S^L and S^R and the activity start time of the activity. The three project schedules can be described along the following lines.

- S^O: The schedule considering no learning effect. Figure 7.3(b) shows the so-called *original schedule* which is the solution found by solving the

DTRTP without activity learning effects (eqs. 7.6 to 7.11), resulting in a project makespan of $C^O_{max} = 48$ time units.

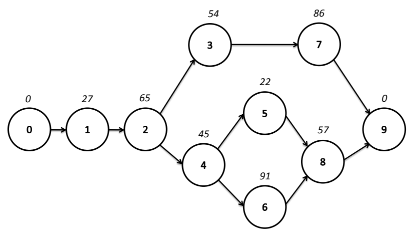
- S^L: The schedule considering learning effect. Figure 7.3(c) shows the so-called *learning schedule* which is the schedule solved by the DTRTP where each activity was subject to activity learning (eqs. 7.6, 7.8-7.10, 7.12), leading to a total project makespan of $C^L_{max} = 49.26$ time units (the introduction of learning effects has led to a makespan increase of more than 2%).

- S^R: The schedule without learning effect using activity duration with learning. Figure 7.3(d) displays the so-called *realistic schedule* where the activity sequence in the schedule has been found by solving the DTRTP without activity learning (eqs. 7.6 to 7.11), but where the activity duration has been replaced afterwards by its learning duration (equation 7.12). The underlying assumption made is that the project manager is not aware of the existence of learning effects during the construction of a baseline schedule and therefore starts executing the project as shown in S^O. However, activities will take more or less time than originally planned due to the existence of learning effects. Therefore, S^R is the schedule where the sequence of the activities is defined by the start times of the different activities in the original schedule S^O, but where the activity durations are changed to the durations where a learning effect is considered. In the example, this results in a project schedule with a makespan of $C^R_{max} = 50.30$ time units. Obviously, this makespan should always be equal to or larger than the C^L_{max}, since in the latter the learning effects have been incorporated in advance.

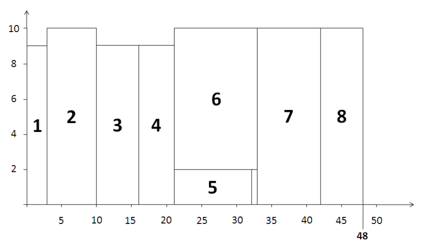
7.4.2 Research design

Minimizing the project duration (makespan) during project scheduling is an important goal in today's competitive industrial environment. In project management, the project baseline schedule is used as a benchmark and point-of-reference during the project's progress. Activity start and finish times are often seen as milestones and are used to follow up the progress of the project. However, the presence of learning effects can dramatically change the project baseline schedule, leading to changes in the activity durations, their start and finish times and consequently the total project makespan.

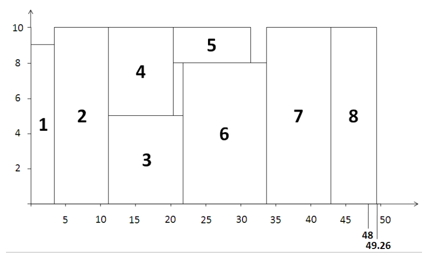
In this section, the impact of learning effects on the project baseline schedule is investigated from three different angles. First, the influence of learning effects on the project makespan is measured to test the relevance and importance of predicting these effects in advance. Second, the influence of learning during project progress is tested to measure the change in the total project makespan when learning effects are ignored during the baseline scheduling phase. Finally, project



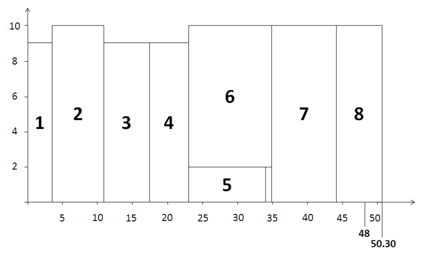
(a) Project network



(b) Schedule S^O



(c) Shedule S^L



(d) Schedule S^R

Act	**Possible modes**	**Chosen mode** (d_{im_i},r_{im_i})			**Start time**		
		S^O	S^L	S^R	S^O	S^L	S^R
0	(0,0)	(0,0)	(0,0)	(0,0)	0	0	0
1	(3,9), (4,7), (5,6), (6,5), (7,4), (9,3)	(3,9)	(3.54,9)	(3.54,9)	0	0	0
2	(7,10), (8,9), (9,8), (10,7), (11,6), (13,5)	(7,10)	(7.38,10)	(7.38,10)	3	3.54	3.54
3	(6,9), (7,8), (8,7), (9,6), (11,5), (14,4)	(6,9)	(10.93,5)	(6.46,9)	10	10.92	10.92
4	(5,9), (6,8), (7,7), (8,6), (9,5), (12,4)	(5,9)	(9.18,5)	(5.51,9)	16	10.92	17.37
5	(3,8), (4,6), (5,5), (6,4), (8,3), (11,2)	(11,2)	(10.93,2)	(10.93,2)	21	20.10	22.88
6	(10,10), (11,9), (12,8), (13,7), (16,6), (19,5)	(12,8)	(11.78,8)	(11.78,8)	21	21.84	22.88
7	(9,10), (10,9), (11,8), (13,7), (15,6), (18,5)	(9,10)	(9.19,10)	(9.19,10)	33	33.63	34.67
8	(6,10), (7,9), (8,8), (9,7), (10,6), (12,5)	(6,10)	(6.46,10)	(6.46,10)	42	42.81	43.85
9	(0,0)	(0,0)	(0,0)	(0,0)	48	49.26	50.30

Figure 7.4: Mode and schedule information for the example project

progress with and without learning effects is compared to reveal the beneficial effect of incorporating learning effects during the early project stages.

In order to analyze these three research questions, the schedules presented in section 7.4.1 are compared to each other. Figure 7.5 graphically displays these research questions, which can be summarized along the following lines.

$$\begin{tikzcd} & S^R \arrow[dl, leftrightarrow, "\text{Error}"'] \arrow[dr, leftrightarrow, "\text{Benefit}"] & \\ S^O \arrow[rr, leftrightarrow, "\text{Impact}"'] & & S^L \end{tikzcd}$$

Figure 7.5: Research design: 3 comparative schedules

1. Impact of learning: The project schedule without learning effects S^O and the schedule with learning effects S^L are compared in order to investigate the *impact* of the introduction of learning effects during the project scheduling phase and to determine the driving variables of the differences between the makespans of both schedules.

2. Margin of error: The proposed schedule S^O is compared to the realistic schedule S^R in order to discover the potential *margin of error* made during project progress (S^R) when the learning effects have been ignored during the project scheduling phase (S^O) but observed afterwards during project progress. The smaller the deviations between both solutions are, the less important it is to spend time and effort to predict the learning effects in advance in order to incorporate them in the project scheduling during baseline schedule construction.

3. Benefits of early knowledge of learning effects: The realistic schedule S^R is compared to the learning schedule S^L in order to measure the *benefits* that can possibly be made when learning effects are detected in early stages of the project progress.

In section 7.5, tests are performed in order to formulate an answer to each of these research topics.

7.4.3 Dataset

In order to test the three research questions of the previous section, a newly created dataset is proposed, containing projects of 15, 20, 25 and 30 activities, each with a work content between 10 and 100. The order strength of the activities varies between 0.10 and 0.90, in steps of 0.10. For every combination 10 projects were generated. In total 4 x 9 x 10 = 360 different project instances were generated.

For the generation of the instances, we have used the RanGen project scheduling instances generator developed by Demeulemeester et al. (2003) and Vanhoucke et al. (2008) and extended the projects to a discrete time/resource trade-off version. The tests are performed with a limited number of modes (2, 3, 4, 5, 6, 7, 8, 9 and 10) and an unlimited number of modes and are generated according to the mode generation rules mentioned above. The resource availability for every renewable resource varies from 10 to 50 in steps of 10. In total, 360 x 10 x 5 = 18,000 different projects were tested and evaluated.

7.5 Computational results

7.5.1 Impact of learning on the project baseline schedule

By making a comparison between the two schedules S^O and S^L, the impact of learning effects on the project makespan can be analyzed and the driving variables can be determined. In order to analyze the influence of the learning rate and the initial efficiency independently, a computational analysis is made for different values of L and $\overline{E}_1$. Both the learning rate L and initial efficiency rate $\overline{E}_1$ varies between 0.7 and 1 in steps of 0.1. For every combination, the C^L_{max} is compared to C^O_{max}. Without loss of generality, it is still assumed that all the resources have the same learning and efficiency values for all the different activities. In total, 18,000 x 4 x 4 = 288,000 projects were analyzed.

Table 7.3 shows the relative deviations between the two schedules which have been measured as follows:

$$dev = \frac{C^L_{max} - C^O_{max}}{C^O_{max}}. \tag{7.13}$$

Next to the influence of the learning rate and the initial efficiency parameters, the influence of other project parameters have also been taken into account but not reported in the table. The results of the tests can be summarized along the following lines.

- Initial efficiency: The initial efficiency has an influence on the project makespan. The higher the initial efficiency, the smaller the makespan. The table shows that for increasing initial efficiency values the average relative deviation decreases from -1.65% to -27.58%, indicating that C^L_{max} becomes smaller and smaller compared to the C^O_{max} and illustrating the beneficial effect of high initial resource efficiencies on the project makespan.

	$L = 70\%$	$L = 80\%$	$L = 90\%$	$L = 100\%$	Average
$E_1 = 70\%$	-36.39%	-19.66%	5.47%	43.98%	**-1.65%**
$E_1 = 80\%$	-41.77%	-27.39%	-6.07%	25.94%	**-12.32%**
$E_1 = 90\%$	-46.13%	-33.58%	-15.22%	11.95%	**-20.75%**
$E_1 = 100\%$	-49.76%	-38.67%	-22.62%	0%	**-27.58%**
Average	**-43.51%**	**-29.82%**	**-9.61%**	**20.65%**	**-15.57%**

Table 7.3: Average relative deviation between C^L_{max} and C^O_{max}

- Learning rate: The learning rate has a similar significant influence on the project makespan. The lower the learning rate value is (i.e. the faster the resources learn), the smaller the makespan with learning C^L_{max} relative to C^O_{max}. Note that the table also shows that a low initial efficiency ($E_1 = 70\%$) and no learning ($L = 100\%$) leads to projects with a much higher makespan than the C^O_{max} (where $E_1 = L = 100\%$). Obviously, in this case, the low initial efficiency is never improved (no learning) which leads to much higher activity duration. The table also shows that, when learning increases (lower learning values), the low efficiencies can be quickly recuperated by these learning effects, leading to improvements up to -36.39%.

- Order strength: Results have shown that the lower the order strength is, the larger the average difference between C^O_{max} and C^L_{max} is. Obviously, in a parallel network (low OS), the S^O will include execution modes with on average longer activity durations than in the serial case, since the activities in the former case can be scheduled in parallel. The fact that the resources become more efficient if they work longer on an activity results in smaller activity durations and thus in a smaller makespan. The inverse is true for serial networks (high OS values) where mainly the modes with short durations are selected which results in an increase of the project duration when learning effects are introduced.

- Resource availability: The resource availability is expressed in an absolute number varying from 10 to 50 in steps of 10. Computational tests have revealed that the average difference between C^O_{max} and C^L_{max} decreases for an increasing resource availability. Obviously, in project settings with a high resource availability level, enough resources are available to execute modes with low durations and hence, the beneficial effects of learning are seldom exploited. The lower the resource availability level becomes, the more modes with on average longer activity durations are chosen, which implies that the efficiency improvements in those cases increase.

- Number of modes: When more mode possibilities can be chosen for each activity, there is more flexibility and room for makespan improvement when learning effects are incorporated. Consequently, the higher the number of modes and activities, the larger the differences between C^O_{max} and C^L_{max} are.

- Average duration: The average duration gives an insight in the average length of an activity. It is calculated for the different mode durations when no learning effects are assumed and can be formulated as $\frac{\sum_{i=1}^{N} \frac{\sum_{j=1}^{M_i} d_{ij}}{M_i}}{N}$. Similar to the number of modes, an increase in the average activity duration increases the average difference between the two project schedules.

Statistical analysis indicates a strong significant influence ($p<0.001$) of the parameters L and $\overline{E}_1$ on the difference between C^O_{max} and C^L_{max}. The other driving variables have a smaller but still significant influence on the direction and the magnitude of the difference.

7.5.2 Margin of error during project progress

In this section, the accuracy of the project schedule S^R is compared to the schedule considering no learning effects (S^O). This research will give insights in the margin of error which is made using the original schedule S^O where learning effects are assumed to be unknown in advance. This project schedule is the baseline schedule proposed to the client and/or is used to set milestones without being aware that the learning effect will occur during project execution. However, efficiency improvements or deficiencies might occur during project progress which affect the activity durations and the total project makespan.

Table 7.4 classifies the set of projects into different deviation classes where the relative deviations are measured as follows:

$$dev = \frac{C^R_{max} - C^O_{max}}{C^O_{max}}. \tag{7.14}$$

Consequently, the higher the deviations, the higher the margin of error is made by ignoring the learning effects during schedule construction. The table shows that only 7.53% of the projects have an absolute deviation smaller than 1%. Most projects (approximately 78%) have a deviation of more than 10%. These tests clearly show that the original baseline schedule S^O is often not an accurate prediction of the real project progress since learning effects will often lead to high deviations between the proposed makespan C^O_{max} and the observed makespan C^R_{max}. Consequently, prior information about the learning effects during the scheduling phase will often generate a competitive advantage in terms of the accuracy of the project schedule and the project makespan promised to the client.

	frequency (%)	cum. frequency (%)
$0 \leq \lvert dev \rvert \leq 0.01$	7.53%	7.53%
$0.01 < \lvert dev \rvert \leq 0.05$	5.71%	13.24%
$0.05 < \lvert dev \rvert \leq 0.10$	8.80%	22.04%
$0.10 < \lvert dev \rvert \leq 0.15$	15.31%	37.35%
$0.15 < \lvert dev \rvert \leq 0.20$	9.68%	47.03%
$0.20 < \lvert dev \rvert \leq 0.25$	9.43%	56.46%
$0.25 < \lvert dev \rvert \leq 0.50$	43.19%	99.65%
$\lvert dev \rvert > 0.50$	0.35%	100.00%

Table 7.4: Frequency table for the relative deviation between C^R_{max} and C^O_{max}

7.5.3 Benefits of early knowledge of learning effects

While the S^R schedule ignores learning effects during the scheduling phase, and observes these effects afterwards during project progress, which leads to changes in the project makespan, the S^L schedule assumes that these learning effects are known from the start and are incorporated during the scheduling phase. However, this section assumes that learning effects can be predicted after some portion of the work is done, and hence, can only be incorporated in the project schedule during project progress by rescheduling the remaining portion of work with the learning information available. Consequently, the realistic schedule S^R can be adapted (i.e. rebaselined) by rescheduling the remaining portion of work with new information obtained during project progress. Therefore, in this section, the rescheduled S^R will be compared with the learning schedule S^L to give insight in the improvements and benefits that can be made when timely incorporating learning effects into the schedule.

In this computational experiment, three rescheduling moments and four rescheduling methods are used. The three rescheduling moments reflect the stages in the project progress where the learning effects are taken into account, and are set to 25%, 50% and 75%. The rescheduling moments assume that learning variables are only known after some time, and eventually lead to a reschedule of the remaining portion of the project schedule S^R while taking the learning effects into account.

Since changes in activity duration/resource modes are not always allowed in the middle of the project due to fixed resource teams, the duration/resource modes of each activity can be modified or can be kept fixed during rescheduling. Moreover, the remaining portion of work can be rescheduled with or without knowledge of learning effects. In doing so, four rescheduling policies are investigated, as summarized along the following lines.

1. Mode change allowed, no learning effect incorporated
2. No mode change allowed, no learning effect incorporated
3. Mode change allowed, learning effect incorporated
4. No mode change allowed, learning effect incorporated

In table 7.5, the average improvement in the project makespan by rescheduling the remaining portion of work is measured as a percentage of the maximum improvement that can be made along the various rescheduling methods and decision moments, as follows:

$$\frac{\sum (C^L_{max} - C^R_{max})}{\sum (C^{new}_{max} - C^R_{max})} \tag{7.15}$$

with C^{new}_{max} the makespan obtained by rescheduling the remaining portion of work of the project including the learning effects. The numerator shows the sum of the absolute makespan deviation of the S^R schedule by rescheduling the remaining portion of work, while the denominator shows the sum of the absolute makespan decrease that can be made by incorporating the learning effects at decision moment 0%. Obviously, C^{new}_{max} is equal to C^L_{max} when the rescheduling moment is set to 0% and the rescheduling method is set to method 3.

The table shows that the rescheduling methods 1 and 2 have a very low or negative impact on the project makespan, as given by the small changes in the ”no learning effect” row. The third and fourth rescheduling methods, however, which reschedule the remaining portion of work with learning effects incorporated, show significant improvements.

The table clearly illustrates that in these cases, a timely incorporation of learning effects leads to bigger improvement. Moreover, the improvements after rescheduling are larger when mode changes are allowed, illustrating that changes in the team member assignments largely affect the efficiency gain which can be obtained. As an example, incorporating learning effects after a quarter of the project progress can lead to 55.80% of the maximum improvement when original team member assignments are allowed to be modified.

	No mode change allowed			Mode change allowed		
	25%	50%	75%	25%	50%	75%
No learning effect	-2.57	-1.67	-1.10	-1.35	2.01	0.83
Learning effect	13.14	8.60	1.96	55.80	37.38	13.38

Table 7.5: Rescheduling methods and decision moments - average improvement (as % of maximum improvement)

7.6 Conclusions

Although empirical evidence is given that learning effects exist in project scheduling environments, the concept is not yet widely applied in the current project scheduling literature. In this chapter, the introduction of learning effects into the discrete time/resource trade-off problem is investigated from various angles. Computational tests found a significant influence of the introduction of learning effects in project scheduling and a practical impact of the concept on management decisions. Computational experiments have been set up to reveal the main project drivers that affect the project makespan when introducing learning effects, to measure the margin of error made by ignoring learning during schedule construction and to show that timely incorporation of learning effects when the project is in progress can lead to significant makespan improvements.

Several future research directions can be suggested. In addition to learning, also forgetting is a natural phenomenon that occurs when a resource stops working on a specific activity. The interruption of the activity obviously leads to the termination of the activity-specific learning process and is extremely significant when activity splitting is allowed. Empirical research has also shown that working with insufficient or too many resources can result in an efficiency decrease due to an increasing loss of motivation and dedication. The introduction of team work where the efficiency of a single resource can influence the team efficiency and the determination of the optimal number of team members and its influence on the project duration is also a topic for further research. Finally, in this research we have focused on the autonomous learning concept. However, the acquired insights in project scheduling with learning effects can be used to discuss and investigate the induced learning concept, where learning is the result of management investments in training and innovative technologies. The determination of the optimal learning rate per activity will be one of the key questions the learning effect research can introduce into the project management practice.

8 Case Study: Audit Scheduling

8.1 Introduction

In this work, several metaheuristic solution procedures for the multi-mode resource-constrained project scheduling problem were proposed. These approaches were applied on theoretical project instances in order to optimize the efficiency of the procedures and to compare the results with other metaheuristics available in the literature. Although our procedures obtained state-of-the-art results on the benchmark datasets, the ultimate goal of our research is to optimize real-life schedules, such that efficiency improvements, cost reductions or profit increases can be realized. In this chapter, we therefore apply one of our metaheuristic approaches on a real-life audit scheduling problem, which consists of generating an appropriate schedule for a small Belgian audit firm.

In audit scheduling, the assignment of a set of auditors to a set of audit tasks, with a predefined arrival time and due date, is optimized. The schedule should specify the audit assignment start and finish time and indicate which auditor will process the task. Several authors have already discussed audit scheduling under different constraints. Chan and Dodin (1986) present a decision support system for audit-staff scheduling. They expand the loading model of Balachandran and Zoltners (1981) in order to deal with precedence constraints among audit tasks, due dates, arrival times, penalty costs for missing audit due dates and the constraint that an auditor cannot process more than one audit task at a time. Furthermore, Dodin and Chan (1991) examine the impact of different objective functions for the audit scheduling problem, while Drexl (1991) presents a hybrid branch and bound-/dynamic programming algorithm to solve an audit scheduling problem in order to minimize the overall costs. Dodin and Elimam (1997) present a procedure for audit scheduling with overlapping activities and sequence-dependent setup costs, such as travel time and cost, while Dodin et al. (1998) tackle this problem with a tabu search procedure. Brucker and Schumacher (1999) also present a tabu search procedure for an audit scheduling problem with the following characteristics: each auditor is only available during disjoint time periods and has a minimal and maximal working time. Moreover, the task of an auditor can be preempted under certain circumstances.

Salewski et al. (1997) state that the above solution procedures are single level models which try to construct a direct assignment of auditors to tasks and periods. Based on a survey among the 200 biggest public accountant firms in Germany, they formulate three levels within audit scheduling: the *medium-term planning*, which assigns teams of auditors to the audit tasks and constructs a schedule by determining the workload per auditor and per week over a planning horizon between three and twelve months; the *medium-to-short-term planning*, which produces a schedule for each auditor that covers all engagements in which he/she is involved in the considered week on the basis of periods of four hours and the *short-term planning*, which assigns the auditors to an audit task in periods of one hour. Salewski et al. (1997) state that the medium-term audit-staff scheduling problem (MASSP) can be formulated as a mode-identity resource-constrained project scheduling problem and determine some priority rules to solve the MASSP. Drexl et al. (2006) propose a column generation method for this problem.

In the remainder of this chapter, an algorithm is presented for the medium-term audit-staff scheduling problem, with sequence-dependent set-up times, varying auditors availability, alternative audit teams and variable audit team efficiencies. This algorithm is applied on real-life data from a small Belgium audit firm, with 15 auditors and almost 250 audit tasks.

This chapter is organized as follows: in section 8.2 the problem formulation is defined, while in section 8.3, a solution approach for this problem will be formulated. The results of a computational study will be discussed in section 8.4, while the conclusions of this chapter are drawn in section 8.5.

8.2 Audit scheduling problem

8.2.1 Description

An audit firm employs a number N of auditors, which have to execute a set E of engagements within a given planning horizon. An engagement includes the execution of a set of audit tasks A. Each audit task of an audit engagement can be performed by different audit teams (modes). The processing time d_{ijm_i} for each task j of engagement i will be different for each audit team m_i and will be influenced by the qualification, industry experience and familiarity with the client's business of each audit team. The number of execution modes for each task within the same engagement is the same.

There may exist a precedence relationship between two audit tasks, which means that the second audit task may not be processed before the first task of the engagement is carried out. Each audit engagement has a time window in which it should be performed: each engagement has a *release time*, which is the contractual date or any time thereafter when the engagement becomes ready for processing and a *due date*, which is the time when the audit engagement should be completed. In most cases this due date is strict, since the due date is a legal restriction of the duration of the audit engagement.

Several audit resource types are available (e.g. senior auditor, assistant auditor or junior auditor) and for each resource type several auditors are available. The availability of some auditors may be restricted in certain periods, e.g. due to holidays, vacations or training. Note that not all audit tasks within the same engagement should be executed by the same audit team. However, some of the tasks should be processed by the same team, due to e.g. legal requirements, while others can be executed by other audit teams. This problem notation refers to the mode identity constraints of Salewski et al. (1997).

We assume that the auditor teams work at a homogeneous efficiency level of 100%. However, every new audit engagement is characterized by an introduction period, in which the audit team learns to know the company, finds out how the company works, etc. This setup time is added to the total duration of the audit task. The idea of setup time incorporation in scheduling is not new and is already studied in the resource-constrained project scheduling (Kaplan, 1991; Kolisch, 1995; Debels and Vanhoucke, 2006; Krüger and Scholl, 2009, 2010) and machine scheduling (Potts and Kovalyov, 2000; Allahverdi et al., 2008). In our problem statement, we assume an audit task-dependent setup time that needs to be added to the audit task at the initial start of the audit engagement as well as for each time a new audit task is performed by another audit team. The setup time can be seen as the loss of time due to the audit team inefficiency at the beginning of a new audit task.

In order to calculate the setup time, we assume that during the introduction period, the auditor teams follow a learning curve, determined by the audit team's initial efficiency $\overline{E}_1$ and learning rate L, until the 100% efficiency level is obtained. The learning variables L and $\overline{E}_1$ give an indication of the audit team's skills: more skilled audit teams have better learning variables (higher initial efficiency $\overline{E}_1$ and lower learning rate L) than less experienced audit teams.

Based on equation 7.3, the *time needed to obtain the 100% efficiency level* (expressed in man-days) is equal to

$$t_s = \sqrt[b]{\frac{100\%}{(1+b)\overline{E}_1}}$$

with $b = -log_2 L$ the learning index depending on the learning rate L. It is assumed that each audit team has a specific learning rate and a specific initial efficiency, which depends on their mixture and level of skills.

The *number of working days performed during the time period* [0,t_s] can be calculated based on equation 7.4 and is equal to

$$\begin{aligned} D^w_{t_s} &= \overline{E}_{i1} t_s^{1+b} \\ D^w_{t_s} &= \overline{E}_{i1} (\sqrt[b]{\frac{100\%}{(1+b)\overline{E}_1}})^{1+b} \end{aligned}$$

Consequently, due to the initial setup period where resources need to reach their 100% efficiency level, t_s man-days will be necessary to perform $D^w_{t_s}$ working days ($t_s \geq D^w_{t_s}$). Hence, the total number of working days performed after d_{ijm_i} man-days can be calculated as the sum of the following two values:

1. $D^w_{t_s}$, which is the number of working days performed during the period [0,t_s] following the efficiency curve determined by the learning variables L and $\overline{E}_1$.

2. $d_{ijm_i} - t_s$, which is the number of working days performed during the period]t_s,d_{ijm_i}], since a homogeneous efficiency (L = $\overline{E}_1$ = 100%) is assumed during this period. In this period, the number of working days is equal to the number of man-days.

The setup time s is defined as the additional time above the predefined d_{ijm_i} man-days (where homogeneous efficiencies are assumed) due to the initial introduction period (in which no homogeneous efficiencies are considered). Hence, the total time is d_{ijm_i} + s man-days. The setup time can be calculated as:

$$\begin{aligned} s &= d_{ijm_i} - [(d_{ijm_i} - t_s) + D^w_{t_s}] \\ s &= t_s - D^w_{t_s} \end{aligned}$$

Consider the example of an audit team with an initial efficiency $\overline{E}_1$ of 0.7 and a learning rate L of 0.85 ($b = 0.234$) executing an activity with a duration d_{ijm_i} of 3 working days. Without setup time, the activity is executed in 3 man-days (due to the homogeneous efficiency), however, when setup times are considered, an introduction period is assumed in which the audit team follows the efficiency curve, determined by the learning variables. It can be calculated that this introduction period takes t_s = 1.86 man-days during which $D^w_{t_s}$ = 1.51 working days are performed (see figure 8.1). The additional duration due to the initial introduction period can be executed in $t_s - D^w_{t_s}$ = 1.86 - 1.51 = 0.35 man-days. This is indicated by the grey area and is equal to the black area, which indicates the loss of efficiency due to learning.

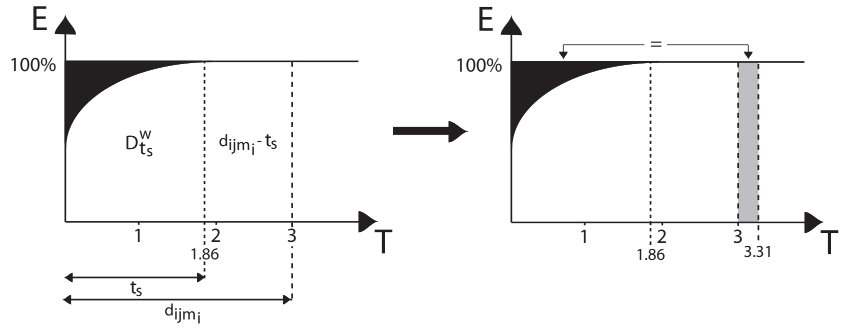


Figure 8.1: Influence of the setup time on audit team switches

With respect to studies published earlier in this research field, some specific elements are not included in the audit scheduling model. First, the minimum and maximum time lags between audit tasks are not taken into account. Second, no preference values are assigned, which means that each execution mode has the same chance to be chosen. Third, no travel times are considered, since we assume that all audits took place at the same place.

8.2.2 Example project

In this section, we present an example for the audit scheduling problem. An overview of the different variables which are used in the remainder of this section is given in table 8.4. The values for the different variables used in this example project are also mentioned in the column 'Example'.

Consider an example project with three audit engagements. Audit engagement 1 and 3 contain 3 audit tasks, audit engagement 2 contains only 2 audit tasks. Each audit engagement has a release date ($\lambda_1 = 0, \lambda_2 = 5, \lambda_3 = 10$) and a due date ($\delta_1 = 10, \delta_2 = \delta_3 = 20$). The planning horizon of the three audit engagements is 20.

Each audit task can be executed in two modes. Each mode represents an auditor team, composed out of one or more auditors. The audit office has in total 5 auditors available, divided into two audit types ($C = 2$): 2 senior auditors and 3 junior

i	m_i	r_{1im_i}	r_{2im_i}	L_{im_i}	$\overline{E}_{1im_i}$	S_{im_i}
1	1	1	2	0.75	0.85	0.19
	2	1	3	0.86	0.73	0.31
2	1	2	3	0.80	0.81	0.20
	2	1	1	0.75	0.90	0.16
	3	0	2	0.77	0.87	0.17
3	1	1	1	0.79	0.84	0.18
	2	0	3	0.84	0.71	0.32

Table 8.1: Audit team: requirements and efficiency measures

i	j	d_{ij1}	d_{ij2}	d_{ij3}	EF_{ij}	LF_{ij}
1	1	2	4	-	2	7
	2	2	3	-	3	8
	3	2	3	-	5	10
2	1	2	4	6	8	13
	2	3	5	7	15	20
3	1	2	3	-	12	16
	2	3	4	-	15	19
	3	1	2	-	16	20

Table 8.2: Audit task durations

i	u_i	H_{iu_i}	h_{iu_i}
1	1	{1,2}	1
	2	{3}	3
2	1	{1}	1
	2	{2}	2
3	1	{1}	1
	2	{2,3}	2

Table 8.3: Mode identity constraints

auditors. The cost of the former is 175 per day, while the cost for the latter is 100 per day. The different audit team combinations are given in table 8.1, including the audit team learning variables L and $\overline{E}_1$. All auditors are available during the complete planning horizon. The duration of the different audit tasks executed by the different audit teams is given in table 8.2. For example, the duration d_{122} of task 2 of engagement 1 executed in mode 2 is 3. This task is executed by an audit team which consists of 1 senior and 3 junior auditors (see mode 2 of engagement 1 in table 8.1). We can also calculate the earliest finish time (EF) and latest finish time (LF) for each audit task, based on the shortest duration for each audit task (without taking the setup time into account).

A mode identity constraint is set to audit engagement 1 and 3. Audit task 1 and 2 of engagement 1 should be executed by the same audit team (i.e. the same mode). This means that if mode 1 is chosen for audit task 1 of engagement 1, also mode 1 has to be chosen for audit task 2 of the same engagement. The same is true for the audit tasks 2 and 3 of engagement 3. In table 8.3, an overview of the different mode identity constraints for the example project is given. The set of audit tasks of each engagement is divided into u_i different subsets and each subset H_{iu_i} contains one or more audit tasks which must be executed by the same audit team. Each audit task is part of exactly one subset ($\sum_{u_i=1}^{U_i} |H_{iu_i}| = A_i, \forall i$).

The setup time of the different audit tasks is defined by the efficiency parameters of the audit team. An overview of the setup times S_{im_i} for the different modes is given in table 8.1. The setup time is applied if the previous audit task is not executed by the same audit team or if the audit task is the first task of the audit engagement. In these cases, the setup time s_{ijm_i} of a specific audit task j of engagement i executed in mode m_i is equal to S_{im_i}. In all other cases, the setup time s_{ijm_i} is equal to 0. For example, the setup time $s_{111} = S_{11} = 0.19$ if the task is executed by mode team 1, while the setup time s_{121} of audit task 2 of engagement 1 is equal to 0 since the task should be executed in the same mode as audit task 1 (see table 8.3).

A precedence relation is set between the different tasks of an audit engagement. The set of predecessors of audit task j of audit engagement i is given by the variable V_{ij}, which consists of elements (o, p), representing an audit task p of an audit engagement o. For example, the element (3,1) is a predecessor of audit task 2 of engagement 3. There is also a precedence relation between audit task 3 of engagement 1 and audit task 1 of engagement 3, since the release time of engagement 3 is equal to the due date of engagement 1 (see also section 8.3.1). So, the element $(1, 3) \in V_{31}$.

In figure 8.2, a feasible solution for the problem is given, taking into account the mode identity constraints, the setup times and the due date restrictions. The chosen audit teams are: team 1 for all tasks in engagement 1, team 1 for audit task 1 and team 2 for audit task 2 of engagement 2 and team 1 for audit task 1 and team

2 for audit task 2 and 3 of engagement 3. The setup times are indicated in grey. The finish time of engagement 1 is 9.20, the finish time of engagement 2 is 12.36 and the finish time of engagement 3 is 18.68.

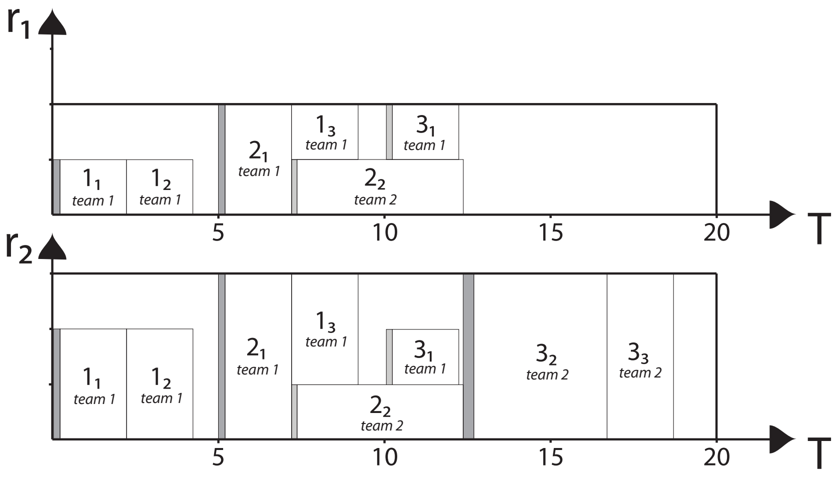


Figure 8.2: Audit scheduling example: a feasible solution

8.2.3 Mathematical formulation

In this section, a mathematical formulation is defined for the audit scheduling problem as defined in the previous section.

The following decision variable is defined:

$$x_{ijm_it} = \begin{cases} 1 & \text{if team } m_i \text{ completes task } j \text{ of engagement } i \text{ at the time } t \\ 0 & \text{otherwise} \end{cases}$$

The model is given by the following equations:

$$\text{optimize } \psi \tag{8.1}$$

$$\sum_{m_i=1}^{M_i} \sum_{t=EF_{ih_{iu_i}}}^{LF_{ih_{iu_i}}} x_{if_{iu_i}m_it} = 1 \qquad 1 \leq i \leq E, 1 \leq u_i \leq U_i \tag{8.2}$$

$$\sum_{t=EF_{ih_{iu_i}}}^{LF_{ih_{iu_i}}} x_{ih_{iu_i}m_it} = \sum_{t=EF_{ij}}^{LF_{ij}} x_{ijm_it}$$

$$1 \leq i \leq E, \forall j \in H_{iu_i} \backslash \{h_{iu_i}\}, 1 \leq u_i \leq U_i, 1 \leq m_i \leq M_i \tag{8.3}$$

General variables		Example variables
T	The planning horizon (index t = 1,...,T)	20
Audit team variables		Example variables
N	The total number of auditors	5
C	The total number of auditor types (index c = 1,...,C)	2
M_i	The number of audit teams which can execute engagement i (index m_i = 1,...,M_i)	$M_1 = M_3 = 2, M_2 = 3$
a_{ct}	The number of auditors of type c which are available at time t	$a_{1t} = 2, a_{2t} = 3, \forall t = 1, ..., T$
L_{ijm_i}	The learning rate for each audit team m_i for audit task j of engagement i	see table 8.1
$\overline{E}_{1ijm_i}$	The initial efficiency rate for each audit team m_i for audit task j of engagement i	see table 8.1
K_c	The cost for one day of audit type c	$K_1 = 175, K_2 = 100$
Audit engagement variables		Example variables
E	The total number of audit engagements (index i = 1,...,E)	3
A_i	The total number of audit tasks for audit engagement i (index j = 1,...,A_i)	A_1 = A_3 = 3, A_2 = 2
d_{ijm_i}	The duration of audit task j of engagement i executed by audit team m_i	see table 8.2
r_{cim_i}	The resource demand of auditor type c of engagement i executed by audit team m_i	see table 8.1
S_{im_i}	The setup time for the different tasks of audit engagement i if executed in mode m_i	see table 8.1
s_{ijm_i}	The setup time of a specific audit task j of audit engagement i if executed in mode m_i	see equation 8.9
δ_i	The due date of audit engagement i	$\delta_1 = 10, \delta_2 = \delta_3 = 20$
λ_i	The release time of audit engagement i	$\lambda_1 = 0, \lambda_2 = 5, \lambda_3 = 10$
U_i	The number of subsets for engagement i (index u_i)	$U_1 = 2, U_2 = 2, U_3 = 2$
H_{iu_i}	The subset of audit tasks for engagement i which must be performed by the same audit team	see table 8.3
h_{iu_i}	The audit task with the smallest audit task index j of the subset H_{iu_i}	see table 8.3
EF_{ij}	The earliest finish time of audit task j of engagement i	see table 8.2
LF_{ij}	The latest finish time of audit task j of engagement i	see table 8.2
F_{ij}	The finish time of audit task j of engagement i	e. g. $F_{13} = 9.20$
V_{ij}	The set of predecessors of audit task j of audit engagement i	e.g. $V_{12} = \{(1, 1)\}$
(o, p)	An element of the set V_{ij} representing an audit task p of an audit engagement o	e.g. $(3, 1) \in V_{32}$

Table 8.4: Variables for the audit scheduling problem

$$\sum_{i=1}^{E}\sum_{m_i=1}^{M_i} r_{cim_i}\sum_{j=1}^{A_i}\sum_{\substack{q=t\\ q\in\{EF_{ij},...,LF_{ij}\}}}^{t+d_{ijm_i}+s_{ijm_i}-1} x_{ijm_iq}\leq a_{ct} \quad 1\leq c\leq C, 1\leq t\leq T \tag{8.4}$$

$$\sum_{t=EF_{op}}^{LF_{op}} tx_{opm_ot}\leq\sum_{t=EF_{ij}}^{LF_{ij}}(t-d_{ijm_i}-s_{ijm_i})x_{ijm_it}$$
$$1\leq i\leq E, 1\leq j\leq A_i, \forall(o,p)\in V_{ij}, 1\leq m_o\leq M_o, 1\leq m_i\leq M_i \tag{8.5}$$

$$\sum_{m_i=1}^{M_i}\sum_{t=EF_{ij}}^{LF_{ij}}(t-d_{ijm_i}-s_{ijm_i})x_{i1m_it}\geq\lambda_i \quad 1\leq i\leq E, 1\leq j\leq A_i \tag{8.6}$$

$$\sum_{m_i=1}^{M_i}\sum_{t=EF_{ij}}^{LF_{ij}} tx_{iA_im_it}\leq\delta_i \qquad 1\leq i\leq E, 1\leq j\leq A_i \tag{8.7}$$

$$x_{ijm_it}\in\{0,1\} \quad 1\leq i\leq E, 1\leq j\leq A_i, 1\leq m_i\leq M_i, EF_{ij}\leq t\leq LF_{ij} \tag{8.8}$$

Equation 8.2 represents that the audit task with the lowest index (audit task h_{iu_i}) of each subset H_{iu_i} is completed exactly once in one of its modes. Equation 8.3 states that each other audit task of the subset should be executed in the same mode as the audit task with the smallest index. In equation 8.4, the resource constraints are set, while in equation 8.5, the precedence relations are defined. Equation 8.6 states that each audit engagement must start on or after its release date λ and equation 8.7 defines that the engagement must finish not later than the due date δ. Finally, equation 8.8 defines the range of decision variables.

In order to incorporate the conditional setup times into the mathematical formulation, the temporary variables $SLK^U_{ijm_i}$ and $SLK^L_{ijm_i}$ are introduced, which represent slack variables and are equal to 0 or 1. The variable w_{ijm_i} represents a boolean variable denoting whether a setup time S_{im_i} should be added to the duration d_{ijm_i} of audit task j of engagement i if executed in mode m_i.

The following equations are added to the mathematical formulation in order to incorporate the conditional setup times:

$$s_{ijm_i} = w_{ijm_i} S_{im_i} \qquad 1 \leq i \leq E, 1 \leq j \leq 1, 1 \leq m_i \leq M_i \qquad (8.9)$$

$$w_{i1m_i} = 1 \qquad 1 \leq i \leq E, 1 \leq m_i \leq M_i \qquad (8.10)$$

$$\sum_{t=EF_{ij}}^{LF_{ij}} x_{ijm_it} - \sum_{t=EF_{ij}}^{LF_{ij}} x_{ij-1m_it} \leq SLK^U_{ijm_i} \quad 1 \leq i \leq E, \forall j > 1, 1 \leq m_i \leq M_i \qquad (8.11)$$

$$\sum_{t=EF_{ij}}^{LF_{ij}} x_{ijm_it} - \sum_{t=EF_{ij}}^{LF_{ij}} x_{ij-1m_it} \leq SLK^L_{ijm_i} \quad 1 \leq i \leq E, \forall j > 1, 1 \leq m_i \leq M_i \qquad (8.12)$$

$$w_{ijm_i} = SLK^L_{ijm_i} + SLK^U_{ijm_i} \quad 1 \leq i \leq E, \forall j > 1, 1 \leq m_i \leq M_i \qquad (8.13)$$

Equation 8.9 defines the value of the setup time of audit task j of engagement i. The value of s_{ijm_i} is equal to 0 or to S_{im_i} and depends on the value of w_{ijm_i}. For the first task of all engagements the value of w_{ijm_i} is equal to 1 (equation 8.10). For all other audit tasks, the value of w_{ijm_i} is equal to 0 if the previous task is executed in the same mode as the current audit task and otherwise equal to 1 (equations 8.11, 8.12 and 8.13).

Different objective functions can be defined. A first objective function minimizes the number of audit engagements finishing after the due date δ.

$$\psi_1 = min \sum_{i=1}^{E} \begin{cases} 1 & F_{iA_i} > \delta_i \\ 0 & \text{otherwise} \end{cases}$$

A second objective function assigns a penalty cost P to each day an audit engagement is finished after the proposed due date δ. If the previous objective function is 0, the result of this objective function will obviously be 0. In this work, the penalty P is, without loss of generality, set at 1 per day.

$$\psi_2 = min \sum_{i=1}^{E} \begin{cases} P(F_{iA_i} - \delta_i) & F_{iA_i} > \delta_i \\ 0 & \text{otherwise} \end{cases}$$

Finally, a third objective function maximizes the value of the remaining resource capacity. The function gives an indication of the resources which are available for other audit engagements and is inspired by the audit office's desire to obtain efficient audit team assignments. The function is calculated as follows:

$$\psi_3 = max \sum_{t=0}^{T} \sum_{c=1}^{C} (a_{ct} - \widehat{a}_{ct}) K_c$$

with $\widehat{a}_{ct}$ equal to the number of resources of resource type c used at time t, as defined in the left hand side of equation 8.4.

8.3 Solution approach

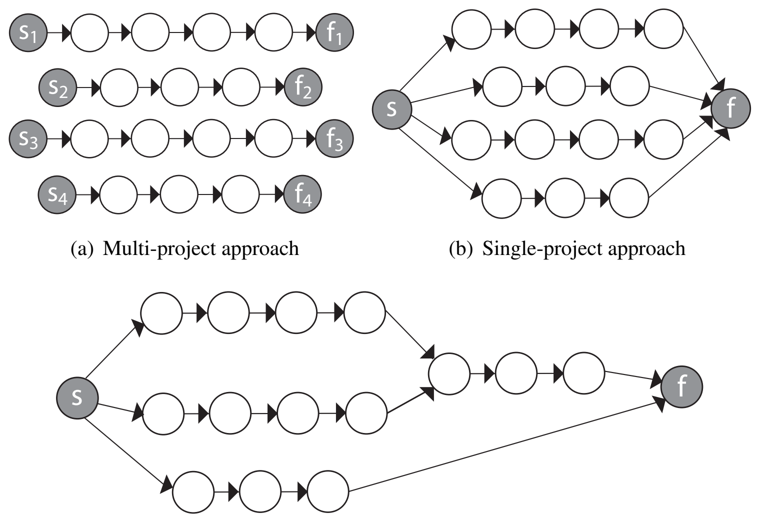
In this section, the solution approach for this audit scheduling problem will be presented. Salewski et al. (1997) proved that the MASSP is a special case of the mode identity resource-constrained project scheduling problem (MIRCPSP). Several adaptations are presented such that the problem can be solved using the genetic algorithm, as proposed in chapter 3. In section 8.3.1, the schedule and solution representation are presented, while in section 8.3.2, a schedule generation scheme with dynamic priority rules is presented in which the mode improvement method is incorporated. Finally, in section 8.3.3, the algorithmic details of the genetic solution procedure are presented.

8.3.1 Representation

Since the audit scheduling problem consists of a set of audit engagements, which are divided in a set of audit tasks between which a precedence relation exists, the problem can be seen as a multiple project scheduling problem, in which a set of single-projects (the audit engagements) should be scheduled sharing the same set of available resource. This can be represented as the different projects shown in figure 8.3(a). However, projects can be bound together artificially into a single project by the addition of two dummy activities representing the start and end of the single aggregate project (Hans et al., 2007), as shown in figure 8.3(b).

Since each audit engagement i has a release date λ_i, before which the engagement cannot start, and a due date δ_i, before which the engagement should be finished due to legal restrictions, extra precedence constraints can be added to the project, without violating the project characteristics. For each pair of engagements (i, j), for which $\delta_i \leq \lambda_j$ is true, a precedence constraint is set between engagements i and j. This results in a project as shown in figure 8.3(c).

Using this representation, the genetic algorithm as proposed in chapter 3 can be used to solve this problem. However, some adaptations are made in order to deal with the specific characteristics of the audit scheduling problem. In the next section, a schedule generation scheme with dynamic priority rules is presented.



(a) Multi-project approach

(b) Single-project approach

(c) Single-project approach with extra precedence constraints

Figure 8.3: Multiple project scheduling problem

8.3.2 Schedule generation scheme with dynamic priority rules

The genetic algorithm makes use of the extended schedule generation scheme with the mode improvement method as presented in chapter 3.3. However, two modifications have been made in order to deal with the specific characteristics of the audit scheduling problem.

1. The *mode improvement method* which selects an activity with a certain probability and evaluates during the generation of the schedule all feasible mode assignments of the selected activity, is used. Due to the mode identity constraints, the improvement method is only applied on the first audit task j of a subset H_{iu_i}. If another mode is chosen for task j, the other audit tasks in set H_{iu_i} also obtain the chosen alternative mode.

2. The sequence in which the activities are scheduled is dynamically updated during the generation of the schedule. To determine this sequence, a priority ϕ_{ij} is assigned to each audit task j of engagement i according to the proximity of the earliest possible start time of the audit task to the engagement due date δ_i. The priority for all eligible audit tasks varies during the generation of the schedule and is calculated as follows:

$$\phi_{ij} = \begin{cases} \delta_i - \lambda_i - d_{ijm_i} & \text{if } j = 1 \\ \delta_i - F_{ij-1} - d_{ijm_i} & \text{otherwise} \end{cases}$$

Audit tasks which approach their due dates have a higher chance to be scheduled with respect to the other eligible activities. Moreover, audit tasks with a larger task duration are also preferred over audit tasks with shorter durations.

The incorporation of this scheduling generation scheme with dynamic priority rules implies that the sequence in which the audit tasks are scheduled is updated dynamically during the generation of the schedule. This implies that the random key, as used in the genetic algorithm proposed in chapter 3 cannot be used. Moreover, tests were also performed using the random key, however, this generation scheme revealed inferior results with respect to the proposed schedule generation scheme with dynamic priority rules.

8.3.3 Algorithmic details

In this section, some specific algorithmic details of the genetic algorithm are presented.

Mode identity Due to the introduction of the mode identity constraint, the crossover and mutation operators are adapted such that a change in mode also leads to the adaption of the mode of the other activities in the set H_{iu_i}.

Crossover As in the genetic algorithm presented in chapter 3, the best results are obtained by using a one-point crossover, which is applied on the mode lists of the population.

Mutation A mutation operator is applied to each mode list. The mutation randomly changes the mode of a randomly selected activity. The probability of an activity to be mutated is equal to 5%.

In the following section, this genetic algorithm is applied on real-life data from a small Belgium audit firm, with 15 auditors and more than 250 audit engagements per year.

8.4 Computational results

In this section, computational tests are performed to test the efficiency of our genetic algorithm, to analyze the influence of the setup costs, mode identity constraints and objective functions and to compare our schedule with the original schedule obtained from the audit firm. In section 8.4.1, the audit firm is presented in detail, while in section 8.4.2, an analysis of three different scenarios is given.

8.4.1 Audit firm

The data used is received from a local Belgian audit firm, which is part of an international network of more than 600 independent offices, present in more than 100 countries. Eight offices are located in Belgium. Each office works independently, but the offices support each other to resolve personnel shortage.

The office, from which the audit scheduling data of the year 2008 is obtained, has an audit department with 15 auditors, divided over 4 audit types: 3 partners, 4 managers, 3 seniors and 5 junior assistants. It is assumed that all auditors of the same type are mutually interchangeable. For the year 2008, the audit firm has obtained 237 audit engagements, each with a duration, a release and a due date. The average engagement duration is 2 weeks, with a minimum of 0.5 day and a maximum of 16 weeks. For each audit engagement, several execution modes are defined, which represent audit teams (composed of one or more auditors) and determine the duration, resource requirements and the audit team efficiency. For each audit engagement, a list of audit tasks which must be executed by the same audit team is available. Since the real cost for the different audit types may not be given, a relative weight is assigned to each audit type. These weights are based on the monetary costs of the different types of auditors. We rescaled the cost of the junior auditor to 100 and compared the cost of any other resource type with this cost. The weight of the junior auditor is therefore set at 100, the weight of the senior assistant is set at 127, the weight of a manager is set at 173 and the weight of a partner is set at 267. The information about the resource availability (holidays, training, ...) is given.

Currently, the audit department makes use of Microsoft Office Excel to manually schedule the audit engagements and tasks. In the remainder of this chapter, we refer to the *original schedule* as the one designed by the audit firm. However, compared to the real-life situation, the following assumptions are made. First, each auditor performs a forty-hour week. This assumption, however, will lead to activities exceeding the engagement due date, although this is not in accordance to the real-life situation in which overtime is used to solve these resource capacity problems. Second, resource inavailability for short periods (e.g. one-day training) is not taken into account since the preemption of activities is not allowed by the algorithm. Obviously, longer periods of unavailability are taken into account.

8.4.2 Analysis

In this section, an evaluation is made for three different scenarios. A first scenario assumes a situation without setup costs and mode identity constraints and is presented in section 8.4.2.1. A second scenario introduces the mode identity constraint and is presented in section 8.4.2.2. Finally, in the third scenario, which is presented in section 8.4.2.3, the setup costs are added to the problem.

		T	ψ_1	ψ_2	ψ_3	switch
Scenario 1	Obj_1	51.13	0	0	265,538	272
	Obj_2	51.13	0	0	267,850	272
	Obj_3	51.13	1	4	270,883	271
Scenario 2	*Original*	51.15	1	4	248,104	150
	Obj_1	51.08	0	0	264,807	144
	Obj_2	51.08	0	0	264,807	144
	Obj_3	51.15	1	31	268,302	128
Scenario 3	*Original*	51.25	2	275	174,395	150
	Obj_1	51.75	2	49	178,964	125
	Obj_2	51.75	2	49	178,964	125
	Obj_3	51.87	7	552	191,848	109

Table 8.5: Results after 5,000 schedules

8.4.2.1 Scenario 1

In this first scenario, the audit scheduling problem is analyzed without taking the mode identity constraint and setup costs into account. The results of this scenario are shown in table 8.5. The problem is optimized according to the three optimization functions (obj_1, obj_2 and obj_3) and is compared to the original schedule. The results for the original schedule under the scenario 1 assumptions is not available, due to the imposed legal restrictions. In the different columns, the objective values (ψ_1 to ψ_3) are mentioned as well as the latest finish time of all activities (T) and the total number of team switches during the execution of the engagements. Schedules for objective functions 1 and 2 obtain feasible schedules, without engagements exceeding the due dates. The schedule obtained for objective function 3 exceeds the due date for 1 engagement for a total of 4 days. The value of the remaining resource capacity is maximized to a total of 270,883.

8.4.2.2 Scenario 2

In the second scenario, the mode identity constraint is introduced. The introduction of this mode identity constraint can be seen as a legal restriction to the change in audit teams during an audit engagement. As can be seen in table 8.5, the number of audit team switches significantly decreases with on average 51% (average switch decrease of obj_1, obj_2 and obj_3). This extra restriction also results in an increase of objective function 1 and 2 and a decrease in objective function 3. The introduction of this legal restriction results in an important efficiency decrease, which results in the decrease of value of the remaining resource capacity.

Compared to the original schedule, the optimization of the mode team assignment, however, leads to an improvement of 8.14% of the value of the remaining work content (268,302 versus 248,104). The introduction of optimization techniques significantly improves the efficiency of the work schedule in terms of auditor assignment.

8.4.2.3 Scenario 3

The same conclusions can be made for the third scenario, in which the setup time is introduced. This setup time can be seen as a switch cost and will affect both the number of audit team switches as well as the number of engagements exceeding the due date, due to the extra time needed to execute an audit task. The setup time is applied every time an audit team switch is applied. In figure 8.4, an example is given of an audit engagement of three audit tasks. Audit team 1, indicated as m_1, executes the first audit task and is faced with an introduction period in which the learning curve is followed. This results in an extra setup time, as explained in section 8.2. This extra time needed to execute audit task 1 is indicated in gray. Since task 2 is also executed by audit team 1, the setup time of this task is equal to 0. The last audit task is performed by audit team 2, which also results in a setup time. Due to the differences in audit team skills, the setup time of audit team 1 is smaller than the setup time of audit team 2. Audit teams with more experienced auditors have better learning variables than teams composed of junior auditors, which results in shorter setup times.

As can be seen in table 8.5, the number of audit team switches decreases significantly under these new settings. Moreover, the number of engagements exceeding the due date and the total number of days exceeding the due date increases significantly.

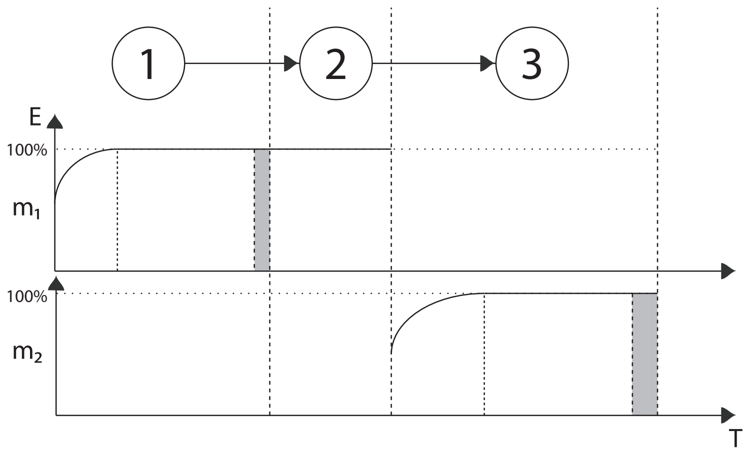
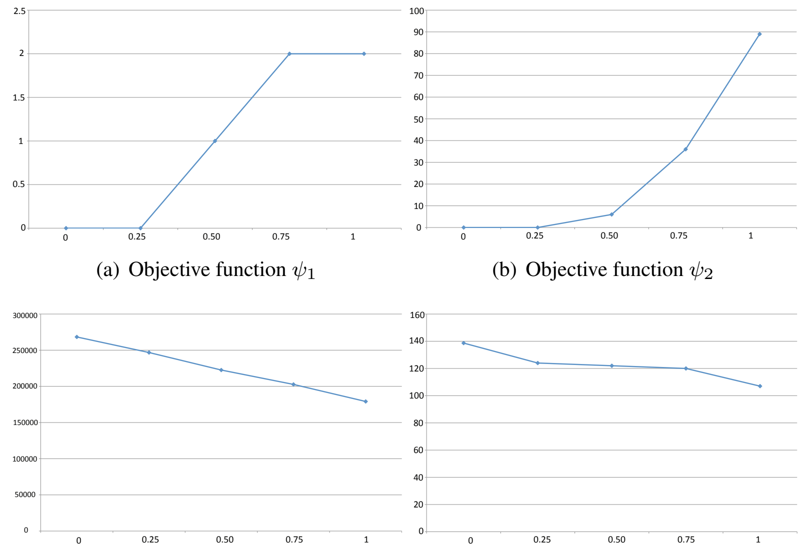


Figure 8.4: Influence of the setup time on audit team switches

In figure 8.5 however, an overview is given for different fixed setup times, varying from 0 to 1 day (in steps of 0.25 days). As can be seen, the values for the objective function ψ_1 and ψ_2 increase for increasing setup times. Moreover, the values for the objective function ψ_2 even increase exponentially. The values for objective function ψ_3 decrease, which is obvious, since increasing setup times leads to the extra use of resource capacity. Finally, the higher the setup time to switch from one audit team to another is, the less audit team switches are made.



(a) Objective function ψ_1 (b) Objective function ψ_2

(c) Objective function ψ_3 (d) Average number of switches

Figure 8.5: Objective function values for different values of the setup time

8.5 Conclusions

In this chapter, a solution is found for a medium-term audit-staff scheduling problem in which the teams of auditors are assigned to a set of audit engagements. Audit team switches are allowed during the execution of an audit engagement, however, mode identity constraints are imposed to some audit tasks, which means that no team switches can be executed due to e.g. legal restrictions. Since an audit team switch also results in an introduction period, in which the audit learns to know the company, an extra setup time is added if an audit team switch is applied.

The introduction of the mode identity constraint has a significant impact on the three objective functions. The efficiency in terms of number of engagements exceeding the due date, the total number of days exceeding the due date and the total value of the remaining resource capacity decrease significantly. The introduction of a setup time further decreases the efficiency, which results in a significant decrease in the number of audit team switches. The same results are obtained for an increasing fixed setup time. The larger the setup time is, the less interesting it is to switch audit teams during audit engagement execution.

The algorithm is applied on real-life data from a small Belgium audit firm, with 15 auditors and almost 250 audit tasks per year. Compared with the optimal schedule obtained in the computational section (result scenario 2 - objective function ψ_3), an efficiency improvement is realized in the audit team assignment of 8.14% in the value of the unassigned resource capacity (from 248,104 to 268,302), which corresponds with almost 200 working days of a junior assistant.

Although the algorithm is applied on a simplification of a practical planning situation, the introduction of optimization techniques significantly improves the efficiency of the work schedule in terms of auditor assignment. As many audit offices still use relatively simple programs to schedule their auditor teams, the use of advanced scheduling algorithms and techniques aims to generate high quality schedules. Future research is needed to develop methods which can incorporate overtime and other real-life extensions and which can convert the obtained audit team schedule into single auditor schedules efficiently.

9 Conclusions and Future Research

In this chapter, an overview of the different research topics of this work is given, the main contribution of each algorithm is analyzed and different directions for future research are defined. Moreover, general reflections are expressed with respect to future research in project scheduling in general and multi-mode resource-constrained project scheduling in particular.

9.1 Introduction

In this work, we investigated the multi-mode resource-constrained project scheduling problem (MRCPSP), where each activity can be performed in different sets of modes, with a specific activity duration and resource requirements. The activities have to be scheduled within precedence constraints and (renewable and nonrenewable) resource constraints, in order to minimize the makespan of the project.

The MRCPSP can be divided into two subproblems: a first subproblem is referred to as the *Mode Assignment Problem (MAP)*, whose aim it is to find a feasible mode assignment. A mode assignment is called feasible if the nonrenewable resource demand does not exceed the nonrenewable resource availability. If there is more than one nonrenewable resource, the problem of finding a feasible solution is NP-complete (Kolisch and Drexl, 1997). In a second subproblem, a start time should be assigned to each activity. Given the duration and the resource consumptions of the different activities, the aim of this scheduling problem is to minimize the makespan of the project.

The resource-constrained project scheduling problem in general and its multi-mode extension in particular have been research topics for many decades, resulting in a wide variety of solution procedures. Different search strategies have been used in order to solve the MRCPSP. An overview of the exact, heuristic and metaheuristic algorithms has been presented in chapter 2.

In Part I of this work, we proposed different population-based metaheuristics. Each of the search strategies has been tailored to the problem under study. The solution procedures have been tested on the PSPLIB dataset (Kolisch et al., 1995) and the dataset of Boctor (1993) and have been compared to other solution procedures available in the literature. Moreover, a new dataset MMLIB was generated in order to deal with the major shortcomings of the current benchmark datasets, given the recent evolution in the development of metaheuristic search procedures. In Part II, we focused on some extensions of the MRCPSP. First, the MRCPSP was extended to the preemptive multi-mode version in order to investigate the influence of preemption on the project duration. Second, the concept of learning was incorporated in the discrete time/resource trade-off problem in order to investigate the influence of the introduction of learning effects on the project makespan. Finally, an audit scheduling problem with sequence-dependent setup times and different audit team efficiencies is modeled as an MRCPSP and the results for a real-life audit team scheduling problem were presented.

The remainder of this chapter is organized as follows. Section 9.2 briefly reviews part I in which three metaheuristic solution procedures for the MRCPSP are introduced. In section 9.3, an overview is given of the two extensions and the case study which are discussed in Part II. Finally, some general reflections are presented in section 9.4.

9.2 Metaheuristic procedures for the MRCPSP

Part I of this work investigated the potential of three different population-based metaheuristic solution procedures to solve the MRCPSP. In section 3.3, a bi-populational genetic algorithm was proposed, which makes use of two populations, one with left-justified schedules and one with right-justified schedules. Also an extended serial schedule generation scheme was proposed, which improves the mode selection by choosing the feasible mode of a certain activity that minimizes the finish time of the activity. In section 3.4, we successfully explored the artificial immune system solution methodology to tackle the MRCPSP. The algorithm makes use of a controlled search procedure, which selects out of a large set of mode lists those lists that have a larger probability to obtain better solutions. This procedure leads the search process more quickly to the more interesting search regions. Finally, the scatter search procedure was proposed in section 3.5. Unlike the other two metaheuristic procedures, the scatter search procedure uses strategic designs to artificially introduce diversity during the search process. The main contribution of this procedure is the steering power of the three proposed improvement methods, each tailored to the specific characteristics of different renewable and nonrenewable resource scarceness values. Moreover, by combining the different improvement methods and two local searches, an efficient combined solution procedure could be designed, which leads to promising computational results. When comparing the performance of the different procedures (cf. chapter 5), we can conclude that the scatter search procedure outperforms the other procedures.

Concerning the design of metaheuristic procedures, we consider the focus on the mode assignment problem and the use of problem specific information in the search process as important directions for future research. The new dataset which has been proposed in chapter 4 can facilitate and motivate researchers to investigate and develop new ideas and techniques to tackle the MRCPSP.

Focus on the mode assignment problem Algorithms using a clever mode assignment procedure to generate the initial population have an advantage compared to other methods. Their performance is better, especially when the number of generated schedules is low. By focusing on the mode assignment problem, i.e. searching for a feasible mode combination, the procedure should be able to reduce the search space, not only in order to exclude infeasible mode assignments, but also to determine those mode combinations with the largest probability of obtaining the optimal solution. Although the use of mode characteristics has already been studied in section 3.4, future research should focus on finding mode parameters or characteristics which influence the makespan of the project.

Use of problem specific information Regardless of the used search strategy, the performance of a metaheuristic is mainly determined by local search procedures used in the algorithm. Procedures which are best tailored to the project settings outperform other procedures. The use of problem specific information in these local search procedures significantly increases the efficiency of the procedure. To that purpose, the design of new local search procedures should be encouraged.

Even though the currently proposed procedures perform very well on the created instances of the benchmark datasets PSPLIB, Boctor and MMLIB, they will probably perform less well on real project instances. Real-life problems will not fit in a specific group of project parameters. In order to provide project managers an efficient schedule, future research should focus on the influence and use of other indicators, not only project specific parameters, but also environmental information and project progress information. A decision support system could provide a link between software packages and the real-life projects and would clearly reduce the gap between academic research and practice.

New dataset for the MRCPSP Based on the disadvantages of the current benchmark datasets PSPLIB and Boctor, we have developed a new dataset MMLIB that has been used to test and validate the proposed metaheuristic solution procedures and to compare the performance of the three algorithms with the metaheuristic solution procedures available in the literature. The introduction of this new dataset opens the possibility to compare new solution procedures with the currently available methods. Researchers are encouraged to use this dataset to compare the results of their solution procedures with other procedures.

9.3 Extensions for the MRCPSP

In part II of this work, we explored two extensions of the MRCPSP, namely the introduction of preemption and the introduction of learning in the MRCPSP. In chapter 6, the influence of preemption on the project duration was investigated. The extension to the preemptive multi-mode version allows activities to be preempted at any integer time instance and restarted later on at no additional cost. This is in contrast with the basic MRCPSP, in which it is assumed that each activity, once started, will be executed until its completion. In order to allow activity preemption, the original activity network is converted into a new network, in which each activity is split into subactivities with a unit duration of 1. The introduction of preemption leads to a significant decrease in the average project makespan compared to the non-preempted case. Nevertheless, it should be stated that it is more difficult to find improvements if nonrenewable resources are taken into account.

In chapter 7, the influence of the introduction of learning effects was investigated. The concept of activity-specific learning, in which the resources became more efficient the longer they stay on the job, was examined from various angles. The concept was introduced in the discrete time/resource trade-off problem, in which each activity contains a specific work content in terms of working days, instead of a fixed duration and resource requirement. For each activity, a set of execution modes can be specified using different combinations of durations and resource requirements, as long as the specified work content is met. Computational tests revealed a significant influence of the introduction of learning effects in project scheduling. The main project drivers that affect the project makespan when introducing learning effects have been analyzed, the margin of error made by ignoring learning during schedule construction has been measured and the importance of incorporating the learning effects timely (i.e. when the project is in progress) has been proven, since this leads to significant makespan improvements.

Nevertheless, with respect to these two extensions, several future research directions can be suggested, as mentioned in the following paragraphs.

Preemption If nonrenewable resources are taken into account, the proposed algorithm was not always able to obtain better solutions compared to the non-preemptive case. This is mainly due to the larger project network, which is generated by splitting the activities into subactivities with a unit duration of 1. New solution procedures must be able to decrease the number of inferior solutions. Future research should therefore focus on an improved mode selection procedure, searching feasible mode assignments more quickly. Another possibility is to limit the number of interruptions allowed during project execution. This probably would have an influence on the results since the search space (here determined as the number of (sub)activities) clearly decreases due to the restricted number of interruptions. Moreover, this restriction proved to be successful in the procedure of Ballestin et al. (2008) for the preempted version of the RCPSP.

In most cases, it is assumed that activities can be preempted without any additional cost. However, this assumption is difficult to maintain in real-life situations. Two types of penalization can be proposed and could be introduced in future research on this topic:

- **Fixed setup time** Every time a preempted activity is restarted, a fixed setup time can be added to the duration of the activity. This penalization is already introduced in the paper of Vanhoucke (2008).

- **Variable setup time** Ash and Smith-Daniels (1999) have determined the impact of learning, forgetting and relearning on the project completion time when preemption is allowed. When an activity is preempted, the efficiency level is interrupted and a period of forgetting is initialized. The variable setup time is determined as the time needed to relearn and obtain the original efficiency level. A similar approach to calculate the setup time is used in chapter 8.

Stochastic durations In chapter 7, the influence of the introduction of learning effects is studied. The influence of learning on the activity durations is measured deterministically. However, a more realistic view could be obtained by considering the learning effect in a stochastic way. The durations should therefore follow a probabilistic distribution, using the formulas given in chapter 7 as the average of the durations. Future research should focus on this point.

In addition to learning, forgetting is also a natural phenomenon that occurs when a resource stops working on a specific activity. The interruption of the activity obviously leads to the termination of the activity-specific learning process and is beneficial when activity splitting is allowed. Empirical research has also shown that working with insufficient or too many resources can result in an efficiency decrease due to an increasing loss of motivation and dedication. The introduction of team work where the efficiency of a single resource can influence the team efficiency and the determination of the optimal number of team members and its influence on the project duration are also topics for further research. Finally, in this research we have focused on the autonomous learning concept. However, the acquired insights in project scheduling with learning effects can be used to discuss and investigate the induced learning concept, where learning is the result of management investments in training and innovative technologies. The determination of the optimal learning rate per activity will be one of the key questions the learning effect research can propose to the project management practice.

Other extensions The general MRCPSP imposes strict assumptions on the activities, which often might be violated in practice. The introduction of preemption and the introduction of the learning concept were two extensions on the general MRCPSP to relax these assumptions. However, other extensions still have to be studied. We refer, amongst others, to the scheduling problem with discounted cash flows or the multi-mode resource availability cost problem. Moreover, also the option to allow within-activity fast tracking should be considered, since this extension proves to be successful in reducing the makespan of schedule (Vanhoucke and Debels, 2008).

Real-life situations In the last chapter of this work, the algorithms designed for the MRCPSP are used to solve real-life scheduling problems, such as the audit team scheduling problem. Although the algorithm is applied on a simplification of a practical planning problem, the introduction of optimization techniques significantly improves the efficiency of the audit team schedule, as can be seen in the computational analysis. Future research, however, is needed to develop methods to incorporate overtime and other real-life extensions. Moreover, other planning problems could also be converted to the MRCPSP and could be solved using the available solution procedures. This will reduce the gap between academic research and practice.

9.4 General reflections

In general, many project scheduling problems are still unexplored. In this work, we have covered the MRCPSP and some of its extensions. To conclude this work, we briefly overview the main contributions of this work.

Three state-of-the-art algorithms In the first part of this work, we have presented three metaheuristic solution procedures. These procedures obtain state-of-the-art results compared to the results of the algorithms available in the literature. Moreover, the use of search strategies that have never been used before to tackle the MRCPSP, such as the scatter search procedure and the artificial immune system method, have proven to be successful in finding high quality solutions.

Use of problem specific information The use of problem specific information in the local search process, such as the use of resource scarceness parameters in the scatter search procedure, increased the efficiency of the procedure significantly.

New dataset The new dataset can facilitate and motivate researchers to investigate and develop new ideas and techniques to tackle the MRCPSP. Researchers are encouraged to use this dataset to compare the results of their solution procedures with other procedures.

Introduction of learning With the introduction of the learning effect in a multi-mode project environment, a new trade-off is set between the duration of an activity and the efficiency of a human resource. We have also learned that the efficiency of human resources influences the project duration. Moreover, the earlier the learning effect is taken into account, the larger the benefit with respect to the original baseline schedule is.

Use for real-life problems The applicability and importance of the multi-mode resource-constrained project scheduling problem are illustrated by means of a real-life audit team scheduling problem bridging the gap between scheduling theory and practice.

References

Aarts, E. and Korst, J. (1989). *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. Wiley, Chichester.

Adler, P. S. and Clark, K. B. (1991). Behind the learning curve: A sketch of the learning process. *Management Science*, 37(3):267–281.

Agarwal, R., Tiwari, M., and Mukherjee, S. (2007). Artificial immune system based approach for solving resource constraint project scheduling problem. *International Journal of Advanced Manufacturing Technology*, 34:584–593.

Alcaraz, J. and Maroto, C. (2001). A robust genetic algorithm for resource allocation in project scheduling. *Annals of Operations Research*, 102:83–109.

Alcaraz, J., Maroto, C., and Ruiz, R. (2003). Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. *Journal of the Operational Research Society*, 54:614–626.

Allahverdi, A., Ng, C., Cheng, T., and Kovalyov, M. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187:985–1032.

Alvarez-Valdes, R. and Tamarit, J. (1989). Heuristic algorithms for resource-constrained project scheduling: A review and emperical analysis. In Slowinski, R. and Weglarz, J., editors, *Advances in Project Scheduling*. Elsevier, Amsterdam.

Amor, J. and Teplitz, C. (1998). An efficient approximation procedure for project composite learning curves. *Project Management Journal*, 29:28–42.

Anderson, E. and Ferris, M. (1994). Genetic algorithm for combinatorial optimisation: The assembly line balancing problem. *ORSA Journal on Computing*, 6:161–173.

Ash, R. and Smith-Daniels, D. E. (1999). The effects of learning, forgetting, and relearning on decision rule performance in multiproject scheduling. *Decision Sciences*, 30:47–82.

Balachandran, B. and Zoltners, A. (1981). An interactive audit-staff scheduling decision support system. *The Accounting Review*, 56:801–812.

Ballestin, F., Valls, V., and Quintanilla, S. (2008). Pre-emption in resource-constrained project scheduling. *European Journal of Operational Research*, 189:1136–1152.

Ballestin, F., Valls, V., and Quintanilla, S. (2009). Scheduling projectswith limited number of preemptions. *Computers and Operations Research*, 36:2913–2925.

Barrios, A., Ballestin, F., and Valls, V. (2009). A double genetic algorithm for the mrcpsp/max. doi:10.1016/j.cor.2009.09.019.

Bedworth, D. and Bailey, J. (1982). *Integrated Production Control Systems - Management, Analysis, Design*. Wiley, New York.

Biskup, D. (2008). A state-of-the-art review on scheduling with learning effects. *European Journal of Operational Research*, 188:315–329.

Blazewicz, J., Lenstra, J., and Rinnooy Kan, A. (1983). Scheduling subject to resource constraints: Classification and complexity. *Discrete Applied Mathematics*, 5:11–24.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308.

Bochenski, B. (1993). *Implementing Production-Quality Client/server Systems*. John Wiley & Sons, Inc.

Boctor, F. (1993). Heuristics for scheduling projects with resource restrictions and several resource-duration modes. *International Journal of Production Research*, 31:2547–2558.

Boctor, F. (1996). A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes. *European Journal of Operational Research*, 90:349–361.

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

Brazel, Y. (1972). The rate of technical progress: The indianapolis 500. *Journal of Economic Theory*, 4:72–81.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., and Pesch, E. (1999). Resource-constrained project scheduling: notation, classification, models, and methods. *European Journal of Operational Research*, 112:3–41.

Brucker, P. and Schumacher, D. (1999). A new tabu search procedure for an audit-scheduling problem. *Journal of Scheduling*, 2(4):157–173.

Buddhakulsomsiri, J. and Kim, D. (2006). Properties of multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. *European Journal of Operational Research*, 175:279–295.

Buddhakulsomsiri, J. and Kim, D. (2007). Priority rule-based heuristic for multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. *European Journal of Operational Research*, 178:374–390.

Chan, K. and Dodin, B. (1986). A decision support system for audit-staff scheduling with precedence constraints and due dates. *The Accounting Review*, 61:726–734.

Coello, C. C., Rivera, D., and Cortes, N. (2003). Use of an artificial immune system for job shop scheduling. *Lecture Notes in Computer Science*, 2787:1–10.

Cooper, D. (1976). Heuristics for Scheduling Resource-constrained Projects: An Experimental Investigation. *Management Science*, 22:1186–1194.

Damay, J., Quilliot, A., and Sanlaville, E. (2007). Linear programming based algorithms for preemptive and non-preemptive rcpsp. *European Journal of Operational Research*, 182:1012–1022.

De Castro, L. and Timmis, J. (2002). Artificial immune systems: a novel paradigm for pattern recognition. In Alonso, L., Corchado, J., and Fyfe, C., editors, *Artificial Neural Networks in Pattern Recognition*. University of Paisley.

De Reyck, B., Demeulemeester, E., and Herroelen, W. (1998). Local search methods for the discrete time/resource trade-off problem in project networks. *Naval Research Logistics*, 45:553–578.

Debels, D., De Reyck, B., Leus, R., and Vanhoucke, M. (2006). A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. *European Journal of Operational Research*, 169:638–653.

Debels, D. and Vanhoucke, M. (2005). A bi-population based genetic algorithm for the RCPSP. *Lecture Notes in Computer Science*, 3483:378–387.

Debels, D. and Vanhoucke, M. (2006). Pre-emptive resource-constrained project scheduling with setup times. Technical report, Ghent University.

Debels, D. and Vanhoucke, M. (2007). A decomposition-based genetic algorithm for the resource-constrained project scheduling problem. *Operations Research*, 55:457–469.

Demeulemeester, E., De Reyck, B., and Herroelen, W. (2000). Discrete time/resource trade-off problem in project networks: A branch-and-bounded approach. *IIE Transactions*, 32:1059–1069.

Demeulemeester, E. and Herroelen, W. (1996). An efficient optimal solution for the preemptive resource-constrained project scheduling problem. *European Journal of Operational Research*, 90:334–348.

Demeulemeester, E., Vanhoucke, M., and Herroelen, W. (2003). A random network generator for activity-on-the-node networks. *Journal of Scheduling*, 6:13–34.

Dodin, B. and Chan, H. (1991). Application of production scheduling methods to external and internal audit scheduling. *European Journal of Operational Research*, 52:267–279.

Dodin, B. and Elimam, A. (1997). Audit scheduling with overlapping activities and sequence-dependent setup costs. *European Journal of Operational Research*, 97:22–33.

Dodin, B., Elimam, A., and Rolland, E. (1998). Tabu search in audit scheduling. *European Journal of Operational Research*, 106:373–392.

Drexl, A. (1991). Scheduling of project networks by job assignment. *Management Science*, 37(12):1590–1602.

Drexl, A., Frahm, J., and Salewski, F. (2006). Audit-staff scheduling by column generation. In Morlock, M., Schwindt, C., Trautmann, N., and Zimmermann, J., editors, *Perspectives on Operations Research*. Gabler Edition Wissenschaft.

Drexl, A. and Grünewald, J. (1993). Nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transactions*, 25:74–81.

Engin, O. and Döyen, A. (2004). A new approach to solve hybrid flow shop scheduling problems by artificial immune system. *Future Generation Computer Systems*, 20:1083–1095.

Geem, Z., Kim, J., and Loganathan, G. (2001). A new heuristic optimization algorithm: Harmony search. *Simulation*, 76:60–68.

Glover, F. and Kochenberger, G. A. (2003). *Handbook of Metaheuristics*. Kluwer Academic Publishers.

Glover, F., Laguna, M., and Marti, R. (2000). Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 29:653–684.

Gutjahr, W. J., Katzensteiner, S., Reiter, P., Stummer, C., and Denk, M. (2008). Competence-driven project portfolio selection, scheduling and staff assignment. *Central European Journal of Operations Research*, 16(3):281–306.

Hanakawa, N., Morisaki, S., and Matsumoto, K.-I. (1998). A learning curve based simulation model for software development. In *20th International Conference on Software Engineering (ICSE'98)*.

Hans, E., Herroelen, W., Leus, R., and Wullink, G. (2007). A hierarchical approach to multi-project planning under uncertainty. *Omega The International Journal of Management Science*, 35:563–577.

Hart, E., Ross, P., and Nelson, J. (1998). Producing robust schedules via an artificial immune system. In *Proceedings of the ICEC '98*.

Hartmann, S. (2001). Project scheduling with multiple modes: A genetic algorithm. *Annals of Operations Research*, 102:111–135.

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

Hartmann, S. and Drexl, A. (1998). Project scheduling with multiple modes: A comparison of exact algorithms. *Networks*, 32:283–297.

Hartmann, S. and Kolisch, R. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 127:394–407.

Hartmann, S. and Sprecher, A. (1996). A note on "hierarchical models for multi-project planning and scheduling". *European Journal of Operational Research*, 94:377–383.

Heimerl, C. and Kolisch, R. (2009). Scheduling and staffing multiple projects with a multi-skilled workforce. *Accepted for publication in OR Spectrum*.

Hendriks, M., Voeten, B., and Kroep, L. (1999). Human resource allocation in a multi-project research and development environment. *International Journal of Project Management*, 17:181–188.

Herroelen, W. and De Reyck, B. (1999). Phase transitions in project scheduling. *Journal of the Operational Research Society*, 50:148–156.

Hillier, F. S. and Lieberman, G. J. (2005). *Introduction to Operations Research*. McGraw-Hill: Boston (MA).

Holland, J. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor.

Homberger, J. (2007). A multi-agent system for the decentralized resource-constrained multi-project scheduling problem. *International Transactions in Operational Research*, 14(6):565–589.

Janiak, A. and Rudek, R. (2007). The learning effect: Getting to the core of the problem. *Information Processing Letters*, 103:183–187.

Jarboui, B., Damak, N., Siarry, P., and Rebai, A. (2008). A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Applied Mathematics and Computation*, 195:299–308.

Józefowska, J., Mika, M., Rózycki, R., Waligóra, G., and Weglarz, J. (2001). Simulated annealing for multi-mode resource-constrained project scheduling. *Annals of Operations Research*, 102:137–155.

Józefowska, J. and Weglarz, J. (2006). *Perspectives in Modern Project Scheduling*. Springer.

Kaplan, L. (1988). *Resource-constrained project scheduling with preemption of jobs*. PhD thesis, University of Michigan.

Kaplan, L. (1991). Resource-constrained project scheduling with setup times. Technical report, Department of Management, University of Tenessee, Knoxville.

Kelley, J. (1963). *The critical-path method: Resources planning and scheduling*. Prentice-Hall, New Jersey.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the IEEE Conference on Neural Networks*.

Knotts, G., Dror, M., and Hartman, B. (2000). Agent-Based Project Scheduling. *IIE Transactions*, 32(5):387–401.

Kolisch, R. (1995). *Project scheduling under resource constraints – Efficient heuristics for several problem classes*. PhD thesis, Physica, Heidelberg.

Kolisch, R. (1996a). Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management*, 14:179–192.

Kolisch, R. (1996b). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 90:320–333.

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

Kolisch, R. and Drexl, A. (1997). Local search for nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transactions*, 29:987–999.

Kolisch, R. and Hartmann, S. (1999). Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis. In Weglarz, J., editor, *Project scheduling: Recent models, algorithms and applications*. Kluwer Academic Publishers.

Kolisch, R. and Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 174:23–37.

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

Krüger, D. and Scholl, A. (2009). A heuristic solution framework for the resource constrained (multi-)project scheduling problem with sequence-dependent transfer times. *European Journal of Operational Research*, 197(2):492–508.

Krüger, D. and Scholl, A. (2010). Managing and modelling general resource transfers in (multi-)project scheduling. *OR Spectrum*, 32(2):369–394.

Li, K. and Willis, R. (1992). An iterative scheduling technique for resource-constrained project scheduling. *European Journal of Operational Research*, 56:370–379.

Lova, A., Tormos, P., and Barber, F. (2006). Multi-Mode Resource Constrained Project Scheduling: Scheduling Schemes, Priority Rules and Mode Selection Rules. *Inteligencia Artificial*, 30:69–86.

Lova, A., Tormos, P., Cervantes, M., and Barber, F. (2009). An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *International Journal of Production Economics*, 117:302–316.

Marti, R., Laguna, M., and Glover, F. (2006). Principles of Scatter Search. *European Journal of Operational Research*, 169:359–372.

Mastor, A. (1970). An experimental and comparative evaluation of production line balancing techniques. *Management Science*, 16:728–746.

Metropolis, N., Rosembluth, A., Rosenbluth, M., and Teller, A. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092.

Mori, M. and Tseng, C. (1997). A genetic algorithm for the multi-mode resource constrained project scheduling problem. *European Journal of Operational Research*, 100:134–141.

Nembhard, D. and Uzumeri, M. (2000). An individual-based description of learning within an organization. *IEEE Transactions on Engineering Management*, 47(3):370 – 378.

Nonobe, K. and Ibaraki, T. (2002). Formulation and tabu search algorithm for the resource constrained project scheduling problem. In Ribeiro, C. and Hansen, P., editors, *Essays and Surveys in Metaheuristics*. Kluwer Academic Publishers.

Osman, I. (1995). An introduction to meta-heuristics. In Lawrence, M. and Wilsdon, C., editors, *Operational Research Tutorial Papers*. Operational Research Society Press.

Osman, I. and Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research*, 63:513–623.

Özdamar, L. (1999). A genetic algorithm approach to a general category project scheduling problem. *IEEE Transactions on Systems, Management and Cybernetics*, 29:44–59.

Özdamar, L. and Ulusoy, G. (1994). A local constraint based analysis approach to project scheduling under general resource constraints. *European Journal of Operational Research*, 79:287–298.

Pascoe, T. (1966). Allocation of resources - CPM. *Revue Française de Recherche Opérationnelle*, 38:31–38.

Patterson, J. (1976). Project scheduling: The effects of problem structure on heuristic scheduling. *Naval Research Logistics*, 23:95–123.

Patterson, J., Slowinski, R., Talbot, F., and Weglarz, J. (1989). An algorithm for a general class of precedence and resource constrained scheduling problem. In Slowinsky, R. and Weglarz, J., editors, *Advances in Project Scheduling*. Elsevier, Amsterdam.

Pinol, H. and Beasley, J. (2006). Scatter Search and Bionomic Algorithms for the Aircraft Landing Problem. *European Journal of Operational Research*, 171:439–462.

PMBOK (2004). *A Guide to the Project Management Body of Knowledge, Third Edition*. Newtown Square, Pa.: Project Management Institute, Inc.

Potts, C. and Kovalyov, M. (2000). Scheduling with batching: A review. *European Journal of Operational Research*, 120:228–249.

Ranjbar, M., De Reyck, B., and Kianfar, F. (2009). A hybrid scatter-search for the discrete time/resource trade-off problem in project scheduling. *European Journal of Operational Research*, 193:35–48.

Ranjbar, M. and Kianfar, F. (2007). Solving the discrete time/resource trade-off problem with genetic algorithms. *Applied Mathematics and Computation*, 191:451–456.

Sahal, D. (1979). A theory of progress functions. *AIIE Transactions*, 11(1):23–29.

Salewski, F., Schirmer, A., and Drexl, A. (1997). Project scheduling under resource and mode identity constraints: Model, complexity, methods and applications. *European Journal of Operational research*, 102:88–110.

Sels, V. and Vanhoucke, M. (2009). A genetic algorithm for the single machine maximum lateness problem. Technical report, Faculty of Economics and Business Administration, Ghent University.

Shtub, A., LeBlanc, L., and Cai, Z. (1996). Scheduling programs with repetitive projects: A comparison of a simulated annealing, a genetic and a pair-wise swap algorithm. *European Journal of Operational Research*, 88:124–138.

Slack, N., Chambers, S., Johnston, R., and Betts, A. (2009). *Operations and process management*. Prentice-Hall, Inc: NJ.

Slowinski, R. (1980). Two approaches to problems of resource allocation among project activities - a comparative study. *Journal of Operational Research Society*, 8:711–723.

Slowinski, R., Soniewicki, B., and Weglarz, J. (1994). DSS for multi-objective project scheduling subject to multiple-category resource constraints. *European Journal of Operational Research*, 79:220–229.

Speranza, M. and Vercellis, C. (1993). Hierarchical models for multi-project planning and scheduling. *European Journal of Operational Research*, 64:312–325.

Sprecher, A. (1994). Resource-constrained project scheduling: Exact methods for the multi-mode case. *Lecture Notes in Economics and Mathematical Systems*, Springer, Berlin.

Sprecher, A. (2000). Scheduling resource-constrained projects competitively at modest memory requirements. *Management Science*, 46:710–723.

Sprecher, A. and Drexl, A. (1998). Multi-mode resource-constrained project scheduling with a simple, general and powerful sequencing algorithm. *European Journal of Operational Research*, 107:431–450.

Sprecher, A., Hartmann, S., and Drexl, A. (1997). An exact algorithm for project scheduling with multiple modes. *OR Sprektrum*, 19:195–203.

Stinson, J., Davis, E., and Khumawala, B. (1978). Multiple Resource-Constrained Scheduling Using Branch-and-Bound. *IIE Transactions*, 10:252–259.

Talbot, F. (1982). Resource-constrained project scheduling problem with time-resource trade-offs: The nonpreemptive case. *Management Science*, 28:1197–1210.

Tavares (1990). A multi-stage non-deterministic model for project scheduling under resource constraints. *European Journal of Operational Research*, 49:92–101.

Tormos, P. and Lova, A. (2001). A competitive heuristic solution technique for resource-constrained project scheduling. *Annals of Operations Research*, 102:65–81.

Tseng, L.-Y. and Chen, S.-C. (2009). Two-phase genetic local search algorithm for the multimode resource-constrained project scheduling problem. *IEEE Transactions on Evolutionary Computation*, 13:848–857.

Upton, D. M. and Kim, B. (1998). Alternative methods of learning and process improvement in manufacturing. *Journal of Operations Management*, 16:1–20.

Valls, V., Ballestin, F., and Quintanilla, S. (2005). Justification and RCPSP: A technique that pays. *European Journal of Operational Research*, 165 (2):375–386.

Valls, V., Ballestin, F., and Quintanilla, S. (2008). A hybrid genetic algorithm for the resource constrained project scheduling problem. *European Journal of Operational Research*, 185(2):495–508.

Valls, V., Laguna, M., Lino, P., Prez, A., and Quintanilla, S. (1999). Project scheduling with stochastic activity interruptions. In Weglarz, J., editor, *Project Scheduling: Recent Models, Algorithms and Applications*. Kluwer Academic Publisher.

Van Peteghem, V. and Vanhoucke, M. (2010). A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problems. *European Journal of Operational Research*, 201:409–418.

Vanhoucke, M. (2008). Setup times and fast tracking in resource-constrained project scheduling. *Computers and Industrial Engineering*, 54:1062–1070.

Vanhoucke, M. (2010). *Measuring Time - Improving Project Performance using Earned Value Management*. International Series in Operations Research and Management Science. Springer.

Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B., and Tavares, L. (2008). An evaluation of the adequacy of project network generators with systematically sampled networks. *European Journal of Operational Research*, 187:511–524.

Vanhoucke, M. and Debels, D. (2008). The impact of various activity assumptions on the lead-time and resource utilization of resource-constrained projects. *Computers and Industrial Engineering*, 54:140–154.

Vanhoucke, M. and Maenhout, B. (2009). On the characterization and generation of nurse scheduling problem instances. *European Journal of Operational Research*, 196:457–467.

Wright, T. (1936). Factors affecting the cost of airplanes. *Journal of Aeronautical Science*, 3:122–128.

Wu, M. and Sun, S. (2006). A project scheduling and staff assignment model considering learning effect. *International Journal of Advanced Manufacturing and Technology*, 28:1190–1195.

Zhang, H., Tam, C., and Li. (2006). Multi-mode project scheduling based on particle swarm optimization. *Computer-Aided Civil and Infrastructure Engineering*, 21:93–103.

Zhu, G., Bard, J., and Tu, G. (2006). A Branch-and-Cut Procedure for the Multimode Resource-Constrained Project-Scheduling Problem. *Journal on Computing*, 18(3):377–390.