

Tilburg University

Airport under Control

Mao, X.

Publication date:
2011

Document Version
Publisher's PDF, also known as Version of record

[Link to publication in Tilburg University Research Portal](#)

Citation for published version (APA):
Mao, X. (2011). *Airport under Control: Multi-agent scheduling for airport ground handling*. TICC Dissertation Series 16.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

AIRPORT UNDER CONTROL

Multiagent Scheduling for Airport Ground Handling

Airport under Control

Multiagent Scheduling for Airport Ground Handling

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Tilburg,
op gezag van de rector magnificus,
Prof. dr. Ph. Eijlander,
in het openbaar te verdedigen ten overstaan van een
door het college voor promoties aangewezen commissie
in de aula van de Universiteit
op woensdag 25 mei 2011 om 10:15 uur

door

Xiaoyu Mao

geboren op 6 december 1980 te Xi'an, P.R. China

Promotores:

Prof. dr. H.J. van den Herik

Prof. dr. E.O. Postma

Copromotores:

Dr. ir. N. Roos

Dr. A.H. Salden

Beoordelingscommissie:

Prof. dr. A.P.J. van den Bosch

Prof. dr. G. van Oortmerssen

Prof. dr. A.J. van Zanten

Prof. dr. C.M. Jonker

Prof. dr. C. Witteveen



Dutch Ministry of Economic Affairs

The research reported in this thesis has been funded by the Dutch Ministry of Economic Affairs in the framework of the Casimir Project program (Project No. CSI4006).



SIKS Dissertation Series No. 2011-13

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.



TiCC Ph.D. Series No. 16

Cover design: Chris Eichberger

ISBN 978-90-5335-401-8

©2011 Xiaoyu Mao

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronically, mechanically, photocopying, recording or otherwise, without prior permission of the author.

Preface

The classical decision theory in project management with a single decision maker soon becomes inapplicable because of the large-scale informational and managerial decentralisation. The rapid change in both technology and the structure of the market place in recent years has called for new paradigms for managing large and distributed projects. Within the field of distributed artificial intelligence, the research area of multiagent systems provide a natural way to model and solve problems with inherent complexity that is caused by large-scale decentralisation.

Our research starts from a practical problem of such a decentralised setting — scheduling airport ground handling (AGH) operations. At an airport, many aircraft are turning around at the same time. Each of the aircraft turnaround processes can be seen as a project involving a multitude of organisations working simultaneously on diverse activities. The general goal of our research is to investigate the characteristics of the AGH scheduling problem and provide an adequate solution model that can solve the problem efficiently and robustly. Our proposed multiagent scheduling system, that is discussed in this thesis, may be used to solve a wider range of real-world scheduling problems.

One of the advantages of doing a PhD at both a university and a research-oriented industrial company is receiving guidance not only from experts in academia, but also from experts in industries. In the academic world, I have had the honour to receive guidance from Jaap van den Herik and Eric Postma, my two supervisors from Tilburg Center for Cognition and Communication (TiCC) at Tilburg University. I owe many thanks to Jaap for his great enthusiasm and support for my research, in particular, for teaching me how to write scientific topics in understandable and attractive texts. A special gratitude goes to Eric for the inspirations he brought into my research. During the early phase of my research, I have had the pleasure to be guided by Nico Roos from Maastricht University. I owe Nico my sincere gratitude for many things. In the industrial world, I am grateful to my daily advisor Alfons Salden from Almende. Alfons always brings me a broader scope of research interests, from fundamental physics to industrial robotics.

Similar to the guidance I received, throughout the whole process of performing research, I received supports and encouragement from many colleagues from Almende, Maastricht University, and Tilburg University. I mention Adriaan ter Mors, Jeroen Valk, Tamás Máhr, Duco Ferro, Anne van Rossum, Andries Stam, Steven de Jong, Jahn Takeshi-Saito, and Laurens van der Maaten. I would like to recognise my Almende colleague Adriaan ter Mors in particular. I have had the pleasure to cooperate with Adriaan in the same research project for four years. Along the way we have built up not only an

enjoyable working partnership but also a life-long valuable friendship.

Moreover, I also wish to acknowledge gratefully the excellent support and help by the management team at Almende and the staff members at Tilburg University. I mention Hans Abbink, Peet van Tooren, Jan Peter Larsen, Judith Engelsman, Janny Ramakers, Joke Hellemons and Olga Houben. I thank Janny in particular for her generous help of translating the english summary into a dutch samenvatting.

In addition, I would like to thank Tony Wauters from KaHo Sint-Lieven for his kindness of sharing his research results in multi-project scheduling.

In conclusion to these acknowledgements, I particularly would like to express my sincere gratitude to my parents. I owe my father eight years of company throughout my oversea life. I thank him for his life-saving financial supports for my Master studies and his weekly moral supports sent from 8,000 kilometres away.

Finally, love to Xiaochen.

Xiaoyu Mao
Rotterdam, May 2011

Contents

Preface	v
Contents	ix
Glossary	xi
List of Figures	xvii
List of Tables	xix
1 Introduction	1
1.1 Airport Ground Handling	2
1.2 Problem Statement and Research Questions	5
1.3 Research Methodology	6
1.3.1 Problem Generalisation and Formulation	6
1.3.2 Literature Review	6
1.3.3 Agent-based Model Design	7
1.3.4 MAS Solutions Development	7
1.3.5 Empirical Evaluation	7
1.4 Structure of the Thesis	7
2 AGH Scheduling Problem	9
2.1 Resource-constrained Project Scheduling Problem	10
2.1.1 Activity and Activity Network	10
2.1.2 Temporal Relations and Constraints	12
2.1.3 Resources and Constraints	16
2.1.4 Schedules and Performance Measures	19
2.2 AGH Scheduling Problem	21
2.2.1 Scheduling Multiple Projects	22
2.2.2 Informational and Managerial Decentralisation	24
2.2.3 Decision Making under Uncertainty	25
2.2.4 A DRCMPSP/u Formulation of AGH Scheduling Problem	27
2.3 Chapter Summary	28

3	A Review of Existing Solution Methods	29
3.1	Solution Methods for RCMPSP	29
3.1.1	Exact Methods	31
3.1.2	Priority-rule-based Heuristics	32
3.1.3	Meta-heuristics	35
3.1.4	Constraint Satisfaction and Optimisation	36
3.1.5	Beyond Centralised Solution Methods	37
3.2	Solution Methods for DRCMPSP	37
3.2.1	Multiagent Systems and Mechanism Design	37
3.2.2	MAS Solutions to DRCMPSP	38
3.2.3	Towards a New MAS Solution	41
3.3	Project Scheduling under Uncertainty	42
3.3.1	Proactive-reactive Scheduling	42
3.3.2	Stochastic Scheduling	43
3.3.3	Fuzzy Scheduling	43
3.3.4	Contingent Scheduling	44
3.3.5	Sensitivity Analysis	44
3.4	Chapter Summary	44
4	A Lease-based Multiagent Model	47
4.1	Agents, Schedules, and Utilities	48
4.1.1	Resource Agent, Schedule, and Utility	49
4.1.2	Project Agent, Schedule, and Utility	52
4.1.3	A Conflict-free and Feasible Agent-based AGH Schedule	55
4.2	Lease-based Market Mechanism	56
4.2.1	Utility Decomposition	57
4.2.2	Lease-based Slot Negotiation	58
4.3	Answer to Research Question 1	64
5	Online Iterative Scheduling	67
5.1	Clairvoyant Online Schedule Generation Scheme	68
5.1.1	Clairvoyant Online Scheme	68
5.1.2	Schedule Generation Schemes	69
5.1.3	An Example	69
5.1.4	A Discussion of Employing COSGS	71
5.2	Iterative Schedule-improvement Method	72
5.2.1	ISIM by Secure-time-window Update	72
5.2.2	ISIM by Resource-type-profile Update	75
5.3	Experiments	78
5.3.1	Experimental Setup	78
5.3.2	Results and Analysis	80
5.4	Answer to Research Question 2	86

6	Stable Proactive Scheduling	87
6.1	Stability: Solution Robustness	87
6.1.1	Stability Measures	88
6.1.2	Stability in Proactive-reactive Scheduling Procedures	89
6.1.3	Solution Models for Stable Proactive Scheduling	90
6.1.4	Towards an Agent-based Stable Scheduling	95
6.2	Agent-based Stable Proactive Scheduling	96
6.2.1	Constructive Heuristic Procedures by Resource Agents	96
6.2.2	Coevolving Slack Time Windows by Project Agents	100
6.3	Experiments	104
6.3.1	Experimental Setup	104
6.3.2	Results and Analysis	105
6.4	Answer to Research Question 3	109
7	Conclusions	111
7.1	Answers to the Research Questions	111
7.1.1	Agent-based Model for AGH Scheduling Problem	111
7.1.2	Efficiency and Robustness under Partial Observability	112
7.1.3	Efficiency and Robustness under Nondeterminism	113
7.2	Answer to the Problem Statement	113
7.3	Recommendations for Future Research	114
	References	117
	Appendices	127
A	Airport Ground-Handling Operations	127
B	Properties of the 80 Chosen MPSPLib Instances	131
	Summary	135
	Samenvatting	139
	Curriculum Vitae	143
	List of Publications	145
	SIKS Dissertation Series	147
	TiCC Ph.D. Series	155

Glossary

List of Abbreviations

The list below contains all technical abbreviations used in the thesis. Normal lexical abbreviations, for instance, ‘e.g.’ and ‘i.e.’, are not listed. Similar considerations apply for organisations, such as BNVKI. Abbreviations used only in tables or figures are explained in the corresponding table or figure.

ADSTW	Activity-dependent Slack Time Window
AGH	Airport Ground Handling
AI	Artificial Intelligence
AoA	Activity on Arc
AoN	Activity on Node
APD	Minimising the Average Project Delay
APDP	Minimising the Average Project Delay Penalty
BPR	Backward Pass Recursion
BSS	Basic Simple Strategy
Co-EAs	Coevolutionary Algorithms
COP	Constraint Optimisation Problem
COS	Clairvoyant Online Scheme
COSGS	Clairvoyant Online Schedule Generation Scheme
CPF	Cohabited Predecessor First
CSP	Constraint Satisfaction Problem
DAI	Distributed Artificial Intelligence
Dec-POMDP	Decentralised Partially Observable Markov Decision Process
DRCMPSP/u	Decentralised Resource-constrained Multi-project Scheduling Problem under Uncertainty
DRCMPSP	Decentralised Resource-constrained Multi-project Scheduling Problem
EFPF	Earliest Finished Predecessor First

EGT	Evolutionary Game Theory
ES	Evolutionary Strategy
FPR	Forward Pass Recursion
GT-MAS	Game-theoretic MAS Scheduling Approach
ISIM	Iterative Schedule-improvement Method
LFT	Minimum Latest Finish Time First
MABO	Myopic Activity-based Optimisation
MAS	Multiagent System
MaxPF	Maximise the Sum of Pairwise Floats
MinEA	Minimise the Number of Extra Arcs
MinED	Minimise the Estimated Disruption
MIP	Mixed Integer Programming
MPSPLib	Library for Multi-project Scheduling Problems
MRCPSP	Multi-mode Resource-constrained Project Scheduling Problem
OI-MAS	Online Iterative MAS Scheduling Approach
OR	Operations Research
p-SGS	Parallel Schedule Generation Scheme
PD	Minimising the Project Delay
PERT	Program Evaluation and Review Technique
PM	Minimising the Project Makespan
PSPLib	Library for Project Scheduling Problems
PS	Problem Statement
RCMPSP	Resource-constrained Multi-project Scheduling Problem
RCPSP	Resource-constrained Project Scheduling Problem
RES	Restart Evolution Strategy
RfQ	Request for Quotation
RPF	Richest Predecessor First
RQ	Research Question
s-SGS	Serial Schedule Generation Scheme
SPD	Minimising the Summed Project Delay
SPM	Minimising the Summed Project Makespan
TCPSP	Time-constrained Project Scheduling Problem
TPM	Minimising the Total Project Makespan
TRCPSP	Time- and Resource-constrained Project Scheduling Problem
TRPC	Minimising the Total Resource Procurement Cost
TSRUC	Minimising the Total Squared Resource Utilisation Cost
TSRU	Minimising the Total Squared Resource Utilisation

List of Symbols

Single-project Scheduling Problem

a_i	i^{th} activity of a project
a_0	Dummy start activity of a project
a_{n+1}	Dummy completion activity of a project
A	Complete set of project (real) activities
A^+	Complete set of project (real and fictitious) activities
\overline{dl}	Project deadline
d_{ij}^{max}	Maximum time lag between the start of activity a_i and a_j
d_{ij}^{min}	Minimum time lag between the start of activity a_i and a_j
\overline{dt}	Project due time
t_i^{ef}	Earliest possible finish time of activity a_i
t_i^{es}	Earliest possible start time of activity a_i
f_i	Scheduled finish time of activity a_i
f_i^*	Actual finish time of activity a_i
\overleftarrow{A}_i	Set of immediate predecessors of activity a_i
\overrightarrow{A}_i	Set of immediate successors of activity a_i
I	A time interval
t_e	End time of a time interval
t_s	Start time of a time interval
t_i^{lf}	Latest possible finish time of activity a_i
t_i^{ls}	Latest possible start time of activity a_i
n	Total number of real activities in a project
p_i	Estimated processing time of activity a_i
\overline{rl}	Expected project release time
s_i	Scheduled start time of activity a_i
S	A complete set of activity start times, a.k.a., a project schedule
s_i^*	Actual start time of activity a_i
\overleftarrow{A}_i^*	Set of transitive predecessors of activity a_i
\overrightarrow{A}_i^*	Set of transitive successors of activity a_i
μ_i	Operating mode of activity a_i
ρ	Length of the project critical path
\prec	Simple finish-start precedence relation

Multi-project Scheduling Problem

$a_{i,j}$	j^{th} activity of project P_i
$a_{i,0}$	Dummy start activity of project P_i
a_{i,n_i+1}	Dummy completion activity of project P_i
A_i	Complete set of all (real) activities in project P_i
A_i^+	Complete set of all (real and fictitious) activities in project P_i
\overline{dl}_i	Deadline of project P_i
\overline{Dl}	Super deadline
\overline{dt}_i	Due time of project P_i
$f_{i,j}$	Scheduled finish time of activity $a_{i,j}$
$f_{i,j}^*$	Actual finish time of activity $a_{i,j}$
$\mu_{i,j}$	Operating mode of activity $a_{i,j}$
n_i	Total number of real activities of project P_i
$p_{i,j}$	Estimated processing time of activity $a_{i,j}$
$p_{i,j}^*$	Actual processing time of activity $a_{i,j}$
P_i	i^{th} project in an RCMPSP
\mathcal{P}	Set of all projects in an RCMPSP
\overline{rl}_i	Expected release time of project P_i
\overline{rl}_i^*	Actual release time of project P_i
\overline{Rl}	Super release time
$s_{i,j}$	Scheduled start time of activity $a_{i,j}$
$s_{i,j}^*$	Actual start time of activity $a_{i,j}$
c_i^{dl}	Delay cost per time unit for project P_i
ρ_i	Length of the critical path of project P_i

Resources

\overline{c}_k	Maximum capacity of resource type R_k
c_k^p	Procurement cost per resource unit of resource type R_k
c_k^u	Utilisation cost per resource unit of resource type R_k
r_i^k	The amount of resources needed by activity a_i from resource type R_k
$r_{i,j}^k$	The amount of resources needed by activity $a_{i,j}$ from resource type R_k
\mathcal{R}	Set of all (renewable) resource types
R_k	k^{th} type of resources
$u_k(S, t)$	Scheduled amount of resource of type R_k to be used at time point t

Agent-based Model for AGH Scheduling

$\Pi_{i,j}$	Schedule of activity $a_{i,j}$
-------------	--------------------------------

$\hat{o}_{i,j,l}$	l^{th} aggregated offer for scheduling $a_{i,j}$
$\hat{O}_{i,j}$	Set of aggregated offers for scheduling $a_{i,j}$
$dl_i(\Pi_i)$	Project delay time given a schedule Π_i
E_k	Set of edged in resource flow network G_k
$f_{v_{i,j}^k \rightarrow v_{i',j'}^k}$	Resource quantity of R_k passing on from activity $a_{i,j}$ to activity $a_{i',j'}$
G_k	Resource flow network of resource type R_k
$dl_i^{mg}(\Pi_i^{\leq i,j})$	Marginal delay caused by the scheduling of activity $a_{i,j}$
$U_{PA_i}^{mg}(\Pi_i^{\leq i,j})$	Marginal project-agent utility of PA_i given a partial schedule $\Pi_i^{\leq i,j}$
$U_{RA_k}^{mg}(\Pi_{\leq i,j}^k)$	Marginal resource-agent utility of RA_k given a partial schedule $\Pi_{\leq i,j}^k$
$o_{i,j,l}^k$	l^{th} offer sent by RA_k for scheduling $a_{i,j}$
$O_{i,j}^k$	Set of offers sent by RA_k for scheduling $a_{i,j}$
PA_i	Project agent representing project P_i
Π_i	Project-agent schedule of project P_i
S^P	Complete set of all project-agent schedules
$U_{PA_i}(\Pi_i)$	Project-agent utility given project-agent schedule Π_i and project unit delay cost c_i^{dl}
$Pr(\pi_{i,j,k,l}^R)$	Price of given resource-agent slot $\pi_{i,j,k,l}^R$
$\pi_{i,j,k}^P$	Project-agent slot
RA_k	Resource agent representing resource type R_k
Π^k	Resource-agent schedule of resource type R_k
$c(\pi_{i,j,k}^R)$	Capacity of resource-agent slot $\pi_{i,j,k}^R$
S^R	Complete set of all resource-agent schedules
$U_{RA_k}(\Pi^k)$	Resource-agent utility given resource-agent schedule Π^k and resource unit utilisation cost c_k^u
$rc(\pi_{i,j,k}^P)$	Resource cost of project-agent slot $\pi_{i,j,k}^P$
$\lambda(\Pi^k, t)$	Resource load of R_k at time t given a resource-agent schedule Π^k
$RfQ_{i,j}^k$	Request for quotations
$\pi_{i,j,k}^R$	Resource-agent slot
$I_{i,j}^s$	Secure time window of $a_{i,j}$
$TC(\hat{o}_{i,j,l})$	Total cost of aggregated offer $\hat{o}_{i,j,l}$
$t_{i,j}^{slk}$	Length of slack time window inserted after activity schedule $\Pi_{i,j}$
t_i^{slk}	Vector of n_i lengths of slack time windows inserted after activities of project P_i
$u_k(\Pi^k, t)$	Amount of R_k scheduled to be used at time t
$v_{i,j}^k$	Vertex representing activity $a_{i,j}$ in resource flow network G_k
V_k	Set of vertices in resource flow network G_k

v_t^k	Sink vertex of resource flow network G_k
v_s^k	Source vertex of resource flow network G_k
$\alpha_{i,j}$	Object variable in (1,1)-ES for activity $a_{i,j}$
$\sigma_{i,j}$	Strategy variable in (1,1)-ES for activity $a_{i,j}$
$\vec{\alpha}_i$	Vector of object variables
$\vec{\sigma}_i$	Vector of strategy variables

List of Figures

1.1	An example of Boeing 747 turnaround schedule in Gantt chart	3
2.1	AoN representation of a project	11
2.2	Allen (1983)'s interval algebra for possible temporal relations	12
2.3	Possible temporal relations between a time point t and a time interval I	13
2.4	Minimum (d_{ij}^{min}) and maximum (d_{ij}^{max}) time lag: $d_{ij}^{min} \leq \alpha \leq d_{ij}^{max}$	13
2.5	Generalised precedence relations by min/max time lags	14
2.6	Refined AoN representation of a project	18
3.1	An example of super AoN network for RCMPSP	30
4.1	Agent encapsulation approaches	49
4.2	A resource-agent slot $\pi_{i,j,k}^R$	50
4.3	An example of a resource-agent schedule with six slots	51
4.4	An activity schedule $\Pi_{i,j}$	54
4.5	Lease-based slot negotiation, step 1 — Sending RfQs	58
4.6	Lease-based slot negotiation, step 2 — Receiving slot offers	59
4.7	Lease-based slot negotiation, step 3 — Aggregating and evaluating slot offers	60
4.8	Lease-based slot negotiation, step 4 — Sending leases requests	61
4.9	Lease-based slot negotiation, step 5 — Making leases	62
4.10	AoN network of an example project P_1	62
4.11	Schedule of $a_{i,j}$ on the timeline of PA_1	64
5.1	AoN network of an example project P_1	69
5.2	Project-agent schedule Π_1 made by the COSGS	71
5.3	The secure time window $I_{1,1}^s$ of $a_{1,1}$ in iteration 1	74
5.4	The secure time window $I_{1,2}^s$ of $a_{1,2}$ in iteration 1	74
5.5	The secure time window $I_{1,3}^s$ of $a_{1,3}$ in iteration 1	74
5.6	Improves project-agent schedule Π_1 using the ISIM	76
5.7	Schedule improvement of $a_{1,2}$ when R_2 profile changes	76
5.8	Project-agent schedule Π_1 in iteration 2	77
5.9	Schedule improvement by ISIM for the 10 projects in I90/10/1	83
5.10	Trade-offs between project-agent objective and resource-agent objective	86
6.1	The AoN networks of two projects (left) and the project schedules (right)	92

6.2	A resource flow network of R_1 and the corresponding resource profile . . .	92
6.3	An alternative resource flow network of R_1 and the resource profile	93
6.4	The AoN network of a project P_1 with a disrupted activity $a_{1,1}$	94
6.5	Schedule Π_1 without slack time	94
6.6	Schedule Π'_1 with a slack time	94
6.7	Two options of obtaining resources for $a_{i,j}$ in a resource flow network . .	97
6.8	Two options of obtaining resources for $a_{i,j}$ in a resource-profile diagram .	97
6.9	The resource flow network and the resource profile diagram of RA_k	98
6.10	Two options of allocating resources for $a_{i,j}$ in resource flow networks . . .	98
6.11	Two options of allocating resources for $a_{i,j}$ in resource-profile diagrams . .	99
6.12	Two options of allocating resources for $a_{i,j}$ in resource flow networks . . .	100
6.13	Two options of allocating resources for $a_{i,j}$ in resource-profile diagrams . .	101
6.14	Probability density function of a beta ($\alpha = 2, \beta = 5$) distribution	105
6.15	1000 samples of actual activity processing duration $p_{i,j}^*$ ($p_{i,j} = 10$)	105
6.16	(1,1)-ES learning curves of the 10 projects in I90/10/1 with a particular instance of incidents	107
6.17	(1,1)-ES learning curves of the 10 projects in I90/10/1 with random in- stances of incidents	108

List of Tables

3.1	Priority-rule-based heuristics	33
3.2	MAS-based solution methods for DRCMPSP	39
4.1	A list of slot offers $O_{i,j}^k$ for scheduling $a_{i,j}$ from RA_k	59
4.2	Evaluating the aggregated offers for scheduling $a_{i,j}$	60
4.3	List of slot offers sent by RA_1	63
4.4	List of slot offers sent by RA_2	63
4.5	Aggregated offers for scheduling $a_{1,1}$	63
5.1	Offers sent by RA_1	70
5.2	Aggregated offer for scheduling $a_{1,1}$	70
5.3	Offers sent by RA_2	70
5.4	Aggregated offers for scheduling $a_{1,2}$	70
5.5	Offers sent by RA_3	71
5.6	Aggregated offers for scheduling $a_{1,3}$	71
5.7	Slot offers sent by RA_1 in the first iteration of the ISIM	73
5.8	Aggregated offers for scheduling $a_{1,1}$ in the first iteration of the ISIM . . .	73
5.9	Slot offers by RA_2 in the first iteration of the ISIM	75
5.10	Aggregated offers for scheduling $a_{1,2}$ in the first iteration of the ISIM . . .	75
5.11	Old offers sent by RA_2	77
5.12	New offers sent by RA_2	77
5.13	ISIM - Resource offers update	77
5.14	Simulated AGH Scheduling Problems	80
5.15	Average improvement ratios by ISIM on the 80 MPSPLib problem instances	81
5.16	Average improvement ratios by ISIM on the simulated AGH instances . .	82
5.17	Project-by-project improvement ratios by ISIM on 10 I90/10 instances . .	82
5.18	Minimal, maximal, and average numbers of iterations to achieve a stable schedule	83
5.19	Comparison of four methods on average project delay (APD)	84
5.20	Comparison of four methods on total squared resource utilisation (TSRU)	85
5.21	Comparison of APD and TSRU on simulated AGH instances	85
6.1	Comparison of three heuristics on total project stability over 100 simulations	106

B.1	Properties of the chosen 80 problem instances from MPSPLib	131
-----	--	-----

Chapter 1

Introduction

In the past decades, globalisation and economic growth have resulted in a worldwide continuous boost of air-traffic demands. Nowadays, the rising flight demands are exceeding the capacities of most existing airports¹. However, the existing airports cannot expand as much as required, because of two significant constraints: *spatial limitations* and *environmental protection regulations* (cf. Graham and Guyer, 1999; Gualandi et al., 2006). Given the constraints, one of the solutions to handle the growing number of flights would be the construction of new major airports and medium-sized airports (EuroControl, 2008). Next to the long-term plan of constructing new airports, making airport operations more efficient also plays an important part to increase current airports' throughputs. Among all airport operations, we are interested in the *ground-handling operations* that are carried out during *aircraft turnaround processes*. Other areas of interest that may increase airport throughputs by supporting reliable turnarounds are the domain of aircraft taxi planning (see ter Mors, 2010) and collaborative operations in improving maintenance contracts (see de Jong, 2010).

An aircraft turnaround process often involves a multitude of organisations working simultaneously on diverse operations. The simultaneous operations are carried out in an environment with a high degree of uncertainty. This makes an aircraft turnaround process time critical. A minor delay in a single operation with respect to one aircraft can create many changes in related work schedules of other operations or even in the schedules of other aircraft. So, a minor delay may lead to a substantial waste of resources. If the occurrence is not anticipated, it may even lead to a large delay of the entire airport.

In addition to the time criticality, the exchange of relevant information is also critical. Different parties involved in a turnaround process have different and often conflicting interests, there will be a limit (possibly legally enforced) to what extent the parties are willing to accommodate their schedules and those of others. A question of a different nature is how much information the parties are willing to exchange.

¹The research for this thesis started in 2005. The economic crisis of 2007-2009 has affected some of our statements in this introductory chapter. In contrast, the commotion with the volcanic ash cloud in April-May 2010 has emphasised the importance of intelligent scheduling and adequate control at airports. All in all, we believe that the economic crisis is temporary. Therefore, we have decided to maintain the original statements, although they should sometimes be read as ideas in the long run.

Scheduling of all aircraft turnaround processes at an airport is a complex task. The inherent problem complexity and environmental uncertainty highlight the challenge of designing a system that can make *efficient* and *robust* schedules. In this thesis, we investigate approaches within a multiagent-system solution framework for designing such a scheduling system.

This introductory chapter starts by providing some background knowledge on the problem domain of our research — the airport ground handling (see Section 1.1). Subsequently, in Section 1.2, the problem statement and three research questions are formulated. This is followed, in Section 1.3, by a description of the research methodology that will be applied to address the research questions and the problem statement. Finally, the structure of the thesis is presented in Section 1.4.

1.1 Airport Ground Handling

Delay is an experience shared by almost anyone who ever travelled. Arguably, it is an inevitable “feature” of any system in the real and often unpredictable world. However, identifying the causes of delays can help the system managers in developing strategies to cope with the disruptions to their plans, and thus improve the system performance.

The CENTRAL OFFICE OF DELAY ANALYSIS² (CODA) within the EUROPEAN ORGANISATION FOR THE SAFETY OF AIR NAVIGATION (EuroControl) is responsible for collecting and analysing information with regard to air-traffic delays in Europe. A recent study of CODA revealed that amongst all causes of aircraft departure delays, airline-related delays are the primary cause; and during 2009, airline-related delays accounted for around 49% of all aircraft departure delays (EuroControl, 2010). Amongst all sources of airline-related delays occurring at airports, ground handling plays a significant role worth to be investigated in more depth (cf. van Leeuwen and Witteveen, 2009). Below, we define *airport ground handling*.

DEFINITION 1.1 Airport Ground Handling (AGH). *Airport ground handling refers to the management of all aircraft turnaround processes at an airport.*

The turnaround process of an aircraft starts when the aircraft lands at an airport and ends when the aircraft takes off for the next flight. During this period of time, which is known as *turnaround time*, a series of ground-handling operations are required for serving the aircraft. Examples of the operations are (re)fuelling, cleaning, catering, passengers handling, and baggage handling. A comprehensive list of aircraft ground-handling operations can be found in Appendix A: Airport Ground-handling Operations.

Figure 1.1 is extracted from the Airport Handling Manual (IATA, 2009) published by INTERNATIONAL AIR TRANSPORT ASSOCIATION (IATA). It shows an example of the turnaround schedule of a Boeing 747 aircraft in Gantt chart (cf. Gantt, 1974). As we see from this chart, such a turnaround process involves diverse ground-handling operations.

Trying to carry out the ground-handling operations simultaneously would reduce turnaround time. This is preferable for three groups: (i) airline companies, (ii) airport authority, and (iii) air passengers. For airline companies, reducing turnaround time will

²In the thesis, we indicate organisation names with small capitals.

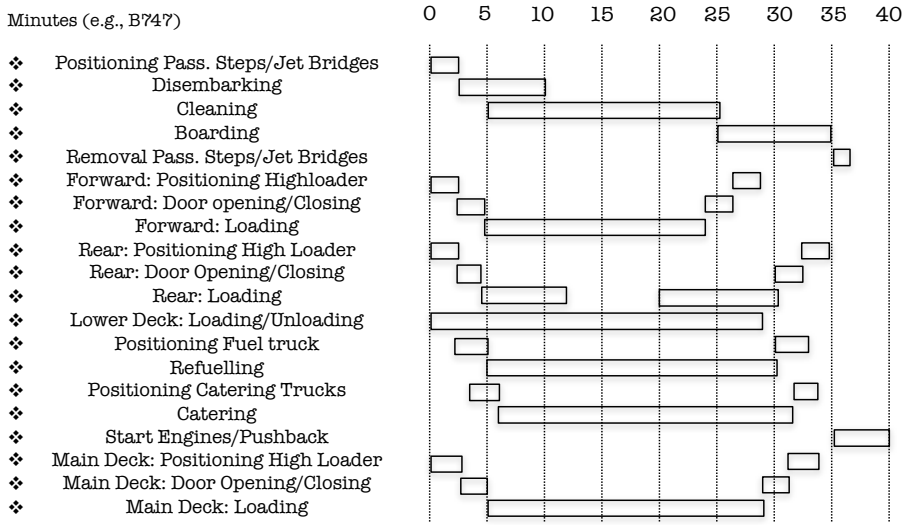


Figure 1.1: An example of Boeing 747 turnaround schedule in Gantt chart

subsequently increase the total flying time of the aircraft and provide the airline companies the opportunity of handling more flights a day, thereby increasing their revenues. Short turnaround time is also advantageous to airport authority, as the use of terminal gates is maximised if turnaround time is kept as short as possible. For air passengers who enjoy punctual aircraft departure and arrival, the efficiency of aircraft turnarounds is the basis of on-time arrival and smooth transit.

However, not all ground-handling operations can be carried out simultaneously. Some of them have to be recorded in a workflow (a sequence of operations) with precedence constraints between one another. A short aircraft turnaround is determined by an efficient planning and scheduling of the ground-handling operations.

Nowadays, planning and scheduling of ground-handling operations in a turnaround process can not be done by one organisation. It is generally not the case that airline companies themselves perform the ground-handling operations for their own aircraft, in particular not when an aircraft is turning around at a remote airport (e.g., an aircraft of Emirates Airline turning around at Amsterdam Schiphol Airport). Many airlines prefer to outsource their remote ground-handling operations either to their alliance partners or to authorised third-party ground-handling parties. In other words, the ground-handling operations are carried out as services provided to the airlines.

In 1996, the EUROPEAN UNION COUNCIL issued a council directive — European Union Ground Handling Council Directive (EU Council, 1996), henceforth the 1996 EU Directive. The objective of the 1996 EU Directive is to encourage the competitive provision of ground-handling services at European airports, in order to (i) reduce airline costs, (ii) improve quality of service, and (iii) provide airlines with the possibility to choose their ground-handling service providers.

The opening up of the AGH market led to a significant increase in the number of

third-party ground-handling service providers at the European airports. Furthermore, it led to free competition on the European AGH market, which lowered the ground-handling prices to the benefit of the airline companies (Airport Research Center, 2009). However, the 1996 EU Directive also led to an ever higher level of complexity and sophistication of AGH management, in particular in relation to the following two aspects.

1. *Coordinated decision making across multiple organisations.*

The liberalisation of the AGH market has resulted in a multitude of organisations involved in a single aircraft turnaround process. Preferably, the ground-handling operations are performed simultaneously to decrease turnaround time. This high degree of simultaneous execution of ground-handling operations requires a high interoperability amongst ground-service providers. The efficiency of such operations relies on (i) the capacity of the staff and technology-advanced equipments of each ground-service provider, and (ii) the coordination amongst the different subcontractors with their own interests and different information support systems. Coordination amongst the different organisations in AGH is in practice carried out by human operators, often connected via (radio) telephones. With the growth of air-transportation volume and the number of organisations, human operators can quickly become overwhelmed by the increased communication and coordination load. The increased need for coordination is a result of the dependencies amongst the plans of the individual organisations, each of which has to adapt its own plans to the joint plan. The limited capacities in inter-human communication and coordination mean that opportunities are missed, and operations slow down, as crucial information does not reach the right actor or planning system or emerges in time.

2. *Dealing with environmental uncertainty.*

The environment of AGH operations is well-known for its large number of disturbances. For instance, the actual arrival time of an aircraft is often different from the one foreseen in the original flight timetable. Uncertainties about the departure time from the departure airport and the duration of a flight result in the uncertainty of the aircraft's arrival time. Moreover, there are uncertainties during the execution of ground-handling operations due to unforeseeable events such as no-show of passengers, breakdown of machinery, and bad weather conditions. As a result, ground-handling operations may take longer time than expected, invalidating the baseline schedule, i.e., a schedule that optimally assigns time and resources to operations under normal conditions. Nowadays, most airports are operating over their normal capacities. This makes the already tightly coupled inter-organisational schedules much tighter. A disturbance by a minor incident may cause a slight change in one aircraft's schedule. However, this slight change may cause a chain of schedule repairs in other aircraft's turnaround schedules, involving a large number of other organisations. Failing to meet the schedule requirements may induce additional costs, which may include resource resetup cost, inventory cost, and various organisational costs.

In summary, the coordinated management of inter-dependent plans and schedules amongst different organisations under uncertainty is an important and complex prob-

lem. We have chosen to investigate this problem in our research. The general problem statement and research questions are formulated in the following section.

1.2 Problem Statement and Research Questions

As described in Section 1.1, scheduling AGH operations involves the coordination of multiple organisations. A global AGH schedule contains all pieces of individual schedules of different organisations. These schedules should respect the individual interests of those organizations.

So far, the scheduling research literature in both *operations research* (OR) and *artificial intelligence* (AI) deals mostly with centralised scheduling problems (cf. Nuijten, 1994; Brucker, 2003; Błażewicz et al., 2007). They assume a central authority in the scheduling system with top-down approaches. However, when designing an AGH scheduling system, one has to take into consideration that the information environment and the managerial decision making are distributed over multiple self-interested organisations.

The conventional centralised scheduling approaches are no longer applicable. The modern way is to distribute the solution process across multiple organisations, following a *distributed artificial intelligence* (DAI) approach (Russell and Norvig, 2003). In particular, *multiagent systems* (MASs), built on the basis of DAI principles, offer a way to understand, manage, and use distributed, large-scale, dynamic, open, and heterogeneous computing and information systems involved in decentralised AGH scheduling.

Some attempts in MAS scheduling have been investigated (see Wellman et al., 2001; Confessore et al., 2007; Homberger, 2007; Wauters et al., 2010). However, these attempts all assume a static and deterministic scheduling environment. The decentralised, dynamic, and nondeterministic scheduling environment in AGH leads us to the following *problem statement* (PS).

PS: *Can a number of self-interested agents, by coordinating their local scheduling decisions, achieve a global AGH schedule that is both efficient and robust?*

From the problem statement above we may derive three specific *research questions* (RQs). First of all, employing a multiagent system as our solution framework calls for an *agent-based model* of the AGH scheduling problem. Thus, we formulate our first research question as follows.

RQ1: *How can an AGH scheduling problem be represented in an agent-based model?*

In general, an agent-based model is composed of (1) a collection of autonomous agents, and (2) inter-agent interactions that lead to emergent properties. Therefore, in order to answer RQ1, two steps have to be taken. First, the roles, characteristics and goals of individual agents have to be specified. Second, the language, protocol, and decision process for inter-agent interactions have to be designed.

The primary objective of coordinating individual agents' decisions through agent interactions is to achieve a global *conflict-free* and *feasible* schedule³. A global schedule

³Definitions of a global conflict-free schedule and a global feasible schedule can be found in §4.1.3.

that is both conflict free and feasible might neither be *efficient* in terms of social welfare, nor be *robust* under uncertainty. In the context of AGH scheduling, uncertainty may encompass many different aspects. In this thesis, we will investigate two classes of uncertainty. They are (i) *partial observability* and (ii) *nondeterminism* (see detailed analyses in §2.2.3). The two different classes of uncertainty lead us to the second and the third research questions.

RQ2: *How can agents make and coordinate their local decisions in order to achieve a globally efficient and robust schedule in a partially observable environment?*

RQ3: *How can agents make and coordinate their local decisions in order to achieve a globally efficient and robust schedule in a nondeterministic environment?*

In the subsequent chapters, we answer the three research questions mentioned above. The answers to the three research questions will allow us to formulate an answer to the problem statement. Below we provide our overall research methodology. The subsequent chapters will describe our approaches in detail.

1.3 Research Methodology

In order to answer the three research questions stated above, we employ an *empirical* research methodology in which we perform the following five main steps: (1) problem generalisation and formulation, (2) literature review, (3) agent-based model design, (4) MAS solutions development, and (5) empirical validation.

1.3.1 Problem Generalisation and Formulation

The thesis aims to design and develop a decentralised scheduling solution framework that not only solves the scheduling problem in AGH, but also covers a wider range of real-world scheduling applications. So, in the first main step, we try (i) to identify the characteristics of the AGH scheduling problem, (ii) to analyse the characteristics of this domain-specific scheduling problem and place the problem in a much broader perspective, and (iii) to reformulate the AGH scheduling problem from a more generic scheduling perspective.

1.3.2 Literature Review

With a generic problem formulation at our disposal, we may conduct a literature review by studying scholarly articles, books, and other sources (e.g., dissertations, industrial reports). We are interested in various techniques and approaches that attempt to solve the generalised scheduling problem. For this purpose we make a description, summary, and critical evaluation of each of a selected number of proposed solution methods in both OR and AI literature. Our goal is to offer an overview of significant contributions from the literature (also articles published on related topics are included) and provide convincing reasons for developing MAS solution methods.

1.3.3 Agent-based Model Design

We model the generic scheduling problem in a heterogeneous MAS solution framework and design a market-based mechanism in which agents are categorised as either *consumer* agents or *producer* agents. All consumer agents have a need of goods produced by the producer agents, and trade or bid for goods at various prices. All agents exchange goods so as to maximise either their profits or their utility. Local decision making amongst agents is coordinated to generate a globally feasible and conflict-free schedule.

1.3.4 MAS Solutions Development

MAS scheduling under uncertainty requires individual agents to make strategic decisions that take into account the dynamics in the environment. In a heterogeneous MAS, different types of agents require different approaches to deal with uncertainty. In addition, we consider two classes of uncertainty — partial observability and nondeterminism. These two uncertainty classes require different scheduling schemes. Accordingly, we design various scheduling schemes and approaches for different types of agents in supporting the efficiency and robustness of the agent decision-making process.

1.3.5 Empirical Evaluation

The last step of our methodology consists of performing a series of experiments. These experiments provide the empirical results that can be used to evaluate the performance of our proposed MAS solution methods within various settings. In general, we conduct two main categories of experiments: (i) scheduling experiments under partial observability and (ii) scheduling experiments under nondeterminism. In each of these two categories, we implement the proposed market-based mechanism in a MAS. The obtained experimental results are used for evaluating the system performance of our solution methods with respect to the conventional OR solution methods (e.g., priority-rule-based heuristic approaches) and centralised AI search methods.

1.4 Structure of the Thesis

The structure of the thesis is as follows.

Chapter 1: Introduction. The chapter introduces the application domain of our research — airport ground handling. A problem statement is formulated and three research questions are derived from the problem statement. In addition, a five-step research methodology is presented.

Chapter 2: AGH Scheduling Problem. In this chapter we identify the characteristics of an AGH scheduling problem and reformulate the problem into a more generic problem definition, viz. that of a project scheduling problem. A formal description of the project scheduling problem is presented and a range of extensions and variations are discussed. We reformulate the AGH scheduling problem as a decentralised resource-constrained multi-project scheduling problem under uncertainty.

- Chapter 3: A Review of Existing Solution Methods.** The chapter reviews the existing solution methods in the literature of project scheduling problems in both OR and AI research. We focus on presenting the state-of-the-art solution methods in solving (1) multi-project scheduling problems, (2) decentralised scheduling problems, and (3) project scheduling under uncertainty. We discuss the limitations of the reviewed solution methods and their (in)applicabilities for solving the AGH scheduling problem. The discussion leads us to a new agent-based model.
- Chapter 4: A Lease-based Multiagent Model.** In this chapter, we propose a novel agent-based model for the AGH scheduling problem. The model adopts a ‘coarse-grained’ physical-entity-oriented modelling approach. It consists of the roles, schedules, and utilities of two classes of agents. We design a market-based coordination mechanism in which the scheduling decisions of the individual agents are coordinated in a lease-based negotiation scenario. The chapter addresses our first research question — RQ1.
- Chapter 5: Online Iterative Scheduling.** The chapter focuses on the first class of AGH scheduling uncertainty — partial observability. We propose an online iterative scheduling approach in the multiagent setting. This approach is composed of (1) a clairvoyant online schedule-generation scheme and (2) an iterative schedule improvement method. By employing this approach, we aim at achieving a globally efficient and robust schedule. Experiments are conducted and empirical analyses are provided to answer RQ2.
- Chapter 6: Stable Proactive Scheduling.** In this chapter, we focus on dealing with the nondeterministic aspect of AGH scheduling problems. We investigate proactive scheduling procedures for constructing stable baseline schedules. In the proactive procedure, two classes of agents employ different approaches (heuristics and evolutionary learning approaches) to construct stable baseline schedules. The constructed schedules should be robust, i.e., being able to tolerate and absorb minor disruptions that may occur during the project execution. A scheduling environment is simulated where the processing times of activities are nondeterministic. The environment is used for evaluating the proposed approaches in dealing with nondeterminism. RQ3 is answered by empirical results.
- Chapter 7: Conclusions.** The chapter concludes the thesis by summarising the answers to the individual research questions and relating them. Moreover, it gives an answer to the problem statement. We also provide a short discussion on potential future research lines.

Chapter 2

AGH Scheduling Problem

The turnaround process of an aircraft consists of a series of ground-handling operations carried out under both temporal and resource constraints. The process can be seen as an instance of a *project* defined in the field of *project management* (Dorndorf, 2002). Project is a broad concept that for different people can refer to many different things. We adopt the concept of *project* used in the context of AGH as follows.

DEFINITION 2.1 Project. *A project is a unique process, consisting of a set of coordinated intermediate activities (or tasks), each of which requires time and resources for its completion. The process is undertaken to achieve one or multiple objectives, while conforming to specific temporal and resource constraints.*

From the project definition above, we may derive the following definition of *project management*.

DEFINITION 2.2 Project Management. *Project management is a set of principles, methods, and technologies applied for the purpose of accomplishing a project (i) on-time, (ii) under budget, and (iii) up to specification.*

Managing a project during its life cycle often involves three phases, namely the phases of *planning*, *scheduling*, and *control* (cf. Lewis, 2005; Kerzner, 2006).

- **Planning** involves defining the project scope (e.g., stakeholders, objectives, and deadline), identifying a work breakdown structure (i.e., a list of intermediate activities and their interdependencies), and estimating the processing duration as well as the resource requirement for each of the intermediate activities.
- **Scheduling** concerns specifying the start times (or the finish times) of all the intermediate activities and allocating the given resources to the activities during their specified time windows.
- **Control** focuses on the difference between the schedule and actual execution once the project has started. During the control phase, project execution is monitored so

that potential problems can be identified in a timely manner and corrective actions can be taken, when necessary.

Our main focus on managing an aircraft turnaround process is the creation of an adequate schedule that establishes start and finish times of the individual operations as well as resource assignment that leads to a successful accomplishment of turnaround process. Therefore, we are interested in the problems that arise in the scheduling phase of the project management and we focus on the development of novel techniques for generating an efficient and robust *schedule*.

In this chapter, we identify the characteristics of an AGH scheduling problem and reformulate the problem within a project-scheduling framework. In Section 2.1, we introduce the classic resource-constrained project scheduling problem (RCPSP) and describe the fundamental concepts within RCPSP. Section 2.2 identifies and discusses the characteristics of the AGH scheduling problem, and formulates the problem as a generalised RCPSP — a decentralised resource-constrained multi-project scheduling problem under uncertainty (DRCMPSP/u). Finally, the chapter is summarised in Section 2.3.

2.1 Resource-constrained Project Scheduling Problem

During the last decades, the resource-constrained project scheduling problem has attracted an ever-growing attention and has become a standard problem for project scheduling in the literature (see Neumann and Zimmermann, 1999; Demeulemeester and Herroelen, 2002). Let us introduce the problem by providing a descriptive definition.

DEFINITION 2.3 Resource-constrained Project Scheduling Problem (RCPSP). *An RCPSP involves the construction of a project schedule that specifies for each activity the start (or finish) time in such a way that the prescribed precedence constraints and resource constraints are satisfied and the objective function(s) is/are optimised.*

In the remainder of the section, we introduce the basic concepts in RCPSP. These include activity and activity network in §2.1.1, precedence relations and constraints in §2.1.2, resources and resource constraints in §2.1.3, and schedules and performance measures in §2.1.4.

2.1.1 Activity and Activity Network

Activities are the essential components of a project. Finishing all activities brings about the completion of the entire project. We assume that a project consists of a set A of $n \in \mathbb{N}$ *real activities*: $A = \{a_1, \dots, a_n\}$, where activity a_i ($a_i \in A$) is to be carried out without interruption¹. Two *fictitious activities* (a *dummy start activity* a_0 and a *dummy completion activity* a_{n+1}) are added to represent *project start* and *project completion*, respectively. Let A^+ denote the set of all activities including the fictitious activities, thus $A^+ = A \cup \{a_0, a_{n+1}\}$. Each activity a_i has an estimated *processing time* (or *duration*) p_i

¹We assume *non-preemptive* activity execution, meaning that once the activity has started, it cannot be interrupted and resumed again. *Preemptive* activities are not the subject of this thesis.

and normally requires resources (except for the dummy activities that consume neither) for execution.

Activities are usually coupled by given dependencies between each other. It is the representation of the dependencies that distinguishes an **activity network** from other ways of representing a project, such as *Gantt chart*, *track planning*, and *line of balance* (Demeulemeester and Herroelen, 2002). There are two possible modes of representing a project using an activity network — the *activity-on-arc* (AoA) representation and the *activity-on-node* (AoN) representation. The latter is more often used since it can represent *generalised precedence relations* (Neumann et al., 2001). More details on generalised activity-to-activity precedence relations are discussed in §2.1.2.

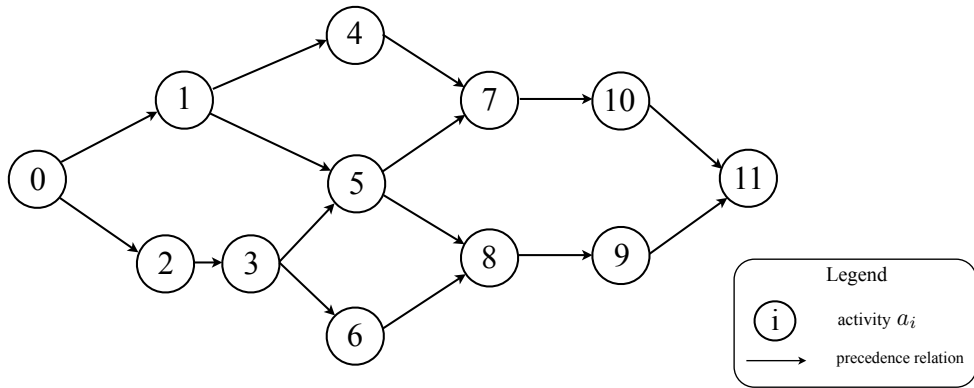


Figure 2.1: AoN representation of a project

The AoN network is a project-network technique often used in project management (cf. Lockyer and Gordon, 2005). In an AoN network, a project is depicted as an *acyclic graph*, consisting of a set of *nodes* representing activities, and a set of *directed arcs* representing *precedence relations* between a pair of activities. The best known precedence relation is *simple finish-start* precedence relation, which tells that for an activity pair (a_i, a_j) , the successor activity a_j can only start (and immediately start) when the predecessor activity a_i has finished. The simple finish-start precedence relation is denoted by $a_i \prec a_j$.

Figure 2.1 shows an example of an AoN network representing a project consisting of 10 real activities. The arcs in Figure 2.1 represent the simple finish-start precedence relations. In an AoN network, nodes are numerically labelled such that the successor nodes always have higher numbers (labels) than all their predecessors.

Below, we define the set of *immediate* predecessors and the set of *immediate* successors of activity a_i . These two sets of activities are denoted by \overleftarrow{A}_i and \overrightarrow{A}_i , respectively.

$$\begin{aligned}\overleftarrow{A}_i &= \{a_j \in A^+ \mid a_j \prec a_i\} \\ \overrightarrow{A}_i &= \{a_j \in A^+ \mid a_i \prec a_j\}\end{aligned}$$

In addition, \overleftarrow{A}_i^* and \overrightarrow{A}_i^* denote the set of *transitive* predecessors and the set of *transitive*

successors of activity a_i , respectively.

$$\begin{aligned}\overleftarrow{A}_i^* &= \overleftarrow{A}_i \cup \overleftarrow{A}_j^*, & \forall a_j \in A^+, a_j \prec a_i \\ \overrightarrow{A}_i^* &= \overrightarrow{A}_i \cup \overrightarrow{A}_j^*, & \forall a_j \in A^+, a_i \prec a_j\end{aligned}$$

For instance, in the example project of Figure 2.1, $\overleftarrow{A}_5 = \{a_1, a_3\}$, $\overrightarrow{A}_5 = \{a_7, a_8\}$, $\overleftarrow{A}_5^* = \{a_0, a_1, a_2, a_3\}$, and $\overrightarrow{A}_5^* = \{a_7, a_8, a_9, a_{10}, a_{11}\}$.

2.1.2 Temporal Relations and Constraints

In this subsection, we deal with the temporal aspects of project scheduling. We start by (a) introducing some basic temporal concepts used in RCPSP. These includes time points, time intervals, and possible temporal relations among them. Then, we (b) discuss how to represent *generalised precedence relations* between a pair of activities in RCPSP. Finally, we (c) discuss several additional temporal constraints.

A: Time Points and Time Intervals

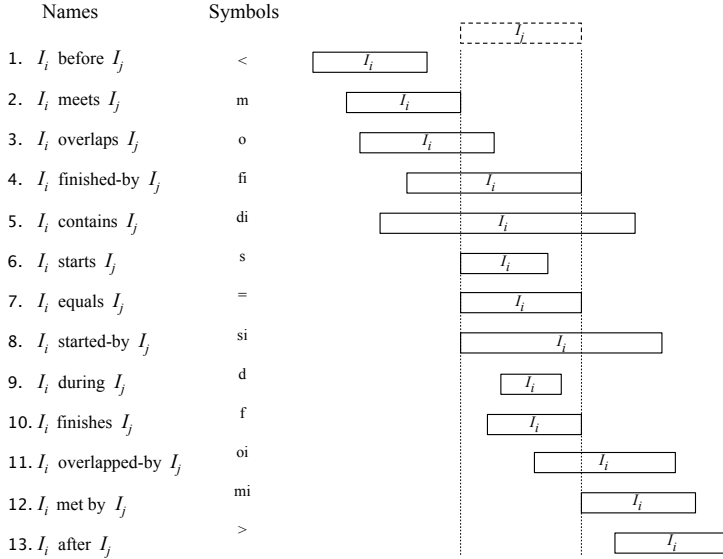


Figure 2.2: Allen (1983)'s interval algebra for possible temporal relations

An important distinction in temporal concepts is that between *time points* and *time intervals* (cf. Vila, 1994). The distinction coincides with the distinction between events and activities in project-scheduling terms. In project scheduling, an event occurs at a point of time, and an activity is occurring over a time interval. Let t denote a time point

and $I = [t_s, t_e)$ denote a time interval that starts at time point t_s (inclusive) and ends at time point t_e (non-inclusive).

According to Allen (1983)'s interval temporal logic, there are thirteen possible temporal relations between a pair of time intervals. These relations are shown in Figure 2.2. By swapping the positions of such a time-interval pair, the number of *interval-to-interval* relations is reduced to seven (see Dorndorf, 2002).

Different from Allen who treated a time point as an indivisible time interval, thus eliminating the need for time points, we opt to maintain the concept of time point in order to address some of the temporal constraints in RCPSP. Since a time interval I is defined by two time points: t_s and t_e , the possible temporal relations between a time point t and a time interval $I = [t_s, t_e)$ (i.e., *point-to-interval* relations) comprise five cases, which is illustrated in Figure 2.3.

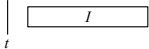
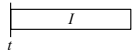
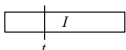
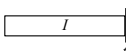
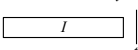
Names	Formula	
1. t before I	$t < t_s$	
2. t starts I	$t = t_s$	
3. t during I	$t_s < t < t_e$	
4. t met-by I	$t = t_e$	
5. t after I	$t_e < t$	

Figure 2.3: Possible temporal relations between a time point t and a time interval I

In addition, we say t is **included by** I (denoted by $t \in I$), when $t_s \leq t < t_e$.

B: Generalised Precedence Relations

In RCPSP, processing an activity requires a time interval. Accordingly, the number of possible temporal relations between a pair of activities (a_i, a_j) is also thirteen. When activity processing times are known and deterministic, we can formulate any of the thirteen temporal relations by using a *start-start* relation with *minimum and maximum time lags*.

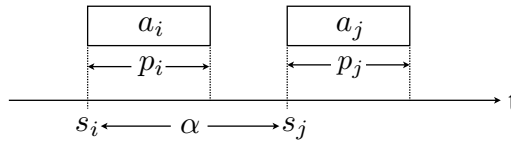


Figure 2.4: Minimum (d_{ij}^{min}) and maximum (d_{ij}^{max}) time lag: $d_{ij}^{min} \leq \alpha \leq d_{ij}^{max}$

Let s_i denote the start time of activity a_i . A given *minimum time lag* $d_{ij}^{min} \in \mathbb{N}$ between the start of two different activities a_i and a_j says that

$$d_{ij}^{min} \leq s_j - s_i. \quad (2.1)$$

That is, activity a_j cannot start earlier than d_{ij}^{min} time units after the start of activity a_i (see Figure 2.4).

If activity a_j can start as soon as activity a_i has finished, i.e., $d_{ij}^{min} = p_i$, inequality 2.1 then represents a simple finish-start precedence constraint as depicted in Figure 2.1.

Moreover, a given *maximum time lag* $d_{ij}^{max} \in \mathbb{N}$ between the start of two different activities a_i and a_j says that

$$s_j - s_i \leq d_{ij}^{max}. \quad (2.2)$$

That is, activity a_j cannot start later than d_{ij}^{max} time units after the start of activity a_i . (see Figure 2.4).

We note that other possible relations, such as *start-finish*, *finish-start*, and *finish-finish* can be trivially transformed into start-start relations when activity processing times are known and deterministic (cf. Dorndorf, 2002). Therefore, a start-start relation with minimum and maximum time lags can represent a *generalised precedence relation* between two different activities (cf. Elmaghraby and Kamburowski, 1992).

Taking relation 2 (i.e., a_i meets a_j) in Figure 2.2 as an example, this relation can be enforced by imposing two constraints $p_i \leq s_j - s_i$ and $s_j - s_i \leq p_i$. Thus, a_i meets a_j can be represented by start-start relation, where $d_{ij}^{min} = d_{ij}^{max} = p_i$.

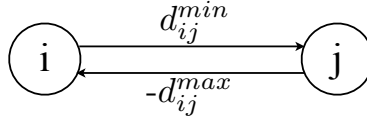


Figure 2.5: Generalised precedence relations by min/max time lags

In an AoN network, one can use bi-directional arrows with minimum and maximum time lags to represent generalised precedence relations, where positive arc weights represent minimum time lags and negative arc weights represent maximum ones (see Figure 2.5). In the thesis, we choose to investigate a simplified precedence relation — simple finish-start precedence relation, and will use one arrow to represent the relation (as shown in Figure 2.1).

C: Additional Temporal Constraints

Apart from activity-to-activity precedence constraints, scheduling a project often has to take into account various additional temporal constraints. In the following, we discuss three of them, in which we face two *hard* constraints: (i) the *project-release-time* constraint, and (ii) the *project-deadline* constraint; and one *soft* constraint: (iii) the *project-due-time* constraint. The hard constraints sometimes are also referred to as “strict” constraints. Violating any of these constraints will cause a project failure. In contrast, soft constraints such as the project-due-time constraint *can be* violated, although it is not a favourable event. Therefore, violating soft constraints often comes with some sort of punishment. Below we discuss all three types of constraints.

i) project-release-time constraint

Project release time is also known as *project arrival time* or *project ready time*. It defines the moment from which the project can be started. Let \overline{rl} denote the project release time. A project-release-time constraint prescribes that no activity of the project can start earlier than \overline{rl} . Since s_0 stands for the start time of the project and all activities start no earlier than s_0 , we may state that

$$\overline{rl} \leq s_0. \quad (2.3)$$

Let t_i^{es} be the *earliest possible start time* and t_i^{ef} be the *earliest possible finish time* of activity a_i ($a_i \in A^+$), respectively ($t_i^{ef} = t_i^{es} + p_i$). Therefore, the earliest possible start time t_0^{es} of the dummy start activity a_0 corresponds to the project release time (i.e., \overline{rl}). During the initialisation step, the earliest possible start times and the earliest possible finish times of all remaining activities can be computed by using the following *Forward Pass Recursion* (FPR) algorithm (see Algorithm 2.1). We recall that \overleftarrow{A}_i denotes the set of immediate predecessors of activity a_i ($a_i \in A^+$).

Algorithm 2.1 Forward Pass Recursion (FPR)

- 1: Initialisation: $t_0^{es} := \overline{rl}, t_0^{ef} := \overline{rl}$
 - 2: **for** $j := 1$ to $n + 1$ **do**
 - 3: $t_j^{es} := \max\{t_i^{ef} | a_i \in \overleftarrow{A}_j\}$
 - 4: $t_j^{ef} := t_j^{es} + p_j$
 - 5: **end for**
-

The FPR algorithm results in the earliest possible start time t_{n+1}^{es} and the earliest possible finish time t_{n+1}^{ef} of the dummy completion activity a_{n+1} . t_{n+1}^{es} is identical to t_{n+1}^{ef} since p_{n+1} is equal to 0. Once t_{n+1}^{ef} is known, the *shortest project duration* or the length of the *project critical path* (denoted by ρ) can be obtained:

$$\rho = t_{n+1}^{ef} - t_0^{es} = t_{n+1}^{ef} - \overline{rl}.$$

We note that t_i^{es} and t_i^{ef} are variables during the course of scheduling process. Their values are updated whenever the schedule of a (transitive) predecessor of a_i is decided or changed.

ii) project-deadline constraint

When a project is given a strict *deadline* \overline{dl} , there is an upper bound on the latest possible project completion time. We note that the project deadline should be greater than the end time of the project critical path: $\overline{dl} \geq \overline{rl} + \rho$, otherwise no feasible schedule exists. A project-deadline constraint can be formulated as follows:

$$s_{n+1} \leq \overline{dl}. \quad (2.4)$$

A *Backward Pass Recursion* (BPR) algorithm yields the latest possible start and finish times of all project activities (see Algorithm 2.2). For activity a_i , the latest possible start time is denoted by t_i^{ls} , and the latest possible finish time is denoted by t_i^{lf} ($t_i^{lf} = t_i^{ls} + p_i$). We recall that \vec{A}_i denotes the set of immediate successors of activity a_i ($a_i \in A^+$).

Algorithm 2.2 Backward Pass Recursion (BPR)

- 1: Initialisation: $t_{n+1}^{lf} := \overline{dl}, t_{n+1}^{ls} := \overline{dl}$
 - 2: **for** $j := n$ to 0 **do**
 - 3: $t_j^{lf} := \min\{t_i^{ls} | a_i \in \vec{A}_j\}$
 - 4: $t_j^{ls} := t_j^{lf} - p_j$
 - 5: **end for**
-

Similar to t_i^{es} and t_i^{ef} , t_i^{ls} and t_i^{lf} are also variables that are updated in the course of the scheduling process, the value of t_i^{ls} and t_i^{lf} are updated whenever the schedule of a (transitive) successor of a_i is decided or changed.

An RCPSP with also a project-deadline constraint is often referred to as the *time- and resource-constrained project scheduling problem* (TRCPSP²) (cf. Neumann et al., 2001).

iii) project-due-time constraint

The project *due time* or *due date* is often set by the project manager(s) during the tactical planning phase of the project management (Hans et al., 2007). A project-due-time constraint is a soft temporal constraint which means that the completion time of a project *can* go beyond its due time, even though this is not favourable. To prevent this from happening, project completion time over its due time is often punished with a penalty referred to as *project delay penalty*.

Let \overline{dt} denote the project due time. \overline{dt} should be set equal to or greater than the end of the project critical path (i.e., $\overline{rl} + \rho$). Thus,

$$\overline{rl} + \rho \leq \overline{dt} \leq \overline{dl}.$$

2.1.3 Resources and Constraints

Project activities require time as well as resources for their executions (the exception being the dummy activities, which are assumed to require no time and no resources). In this subsection, we discuss the resource aspects of project scheduling. These include (a) resource categories, (b) activity operating modes, and (c) resource-capacity constraints.

A: Resource Categories

The resources for carrying out project activities may be of different categories. In general, resources can be divided into three categories: *renewable resources*, *non-renewable*

²Guldemon et al. (2008) use the term TCPSP for a class of project-scheduling problems, where additional temporal constraints are introduced on the activity level, such as a release time or a deadline for each activity.

resources, and *doubly-constrained resources* (Błażewicz et al., 1983). Below we define them.

DEFINITION 2.4 Renewable resource. *Renewable resources are those resources available on a period-by-period basis. The amount of resources is renewable from period to period, only the total resource used at every time instant is constrained.*

Typical examples of renewable resources include manpower, machines, tools, equipment, space, etc.

DEFINITION 2.5 Non-renewable resource. *Non-renewable resources are those resources available on a total project basis, with a limited consumption availability for the entire project.*

The best examples of non-renewable resources are money and energy.

DEFINITION 2.6 Doubly-constrained resource. *Doubly-constrained resources are those resources constrained per period as well as for the overall project.*

Doubly-constrained resources can be incorporated by a combination of renewable and non-renewable resources. Examples are: (i) capitals with a restricted period of cash flow and a limited total of cash amount, and (ii) man-hours per day in combination with a constraint on the total number of man-hours for the entire project.

In this thesis, we focus on the study of renewable resources in RCPSP, and refrain from studying non-renewable resources and doubly-constrained resources. For more information on the latter topics, we refer the readers to Servakh and Shcherbinina (2007).

B: Activity Operating Modes

Within the category of renewable resources, there are also various resource types. In order to carry out a project, different types of renewable resources are often needed. We assume that a set \mathcal{R} of K renewable resource types, $\mathcal{R} = \{R_1, \dots, R_K\}$, is required for carrying out the activities of the project in question. In the project planning phase, the project manager must decide for each activity a_i , (i) the resource requirement, which includes the required resource type(s) and the corresponding amount of each type needed for carrying out the activity³: $\{(k: r_i^k) | k \in \{1, \dots, K\} \wedge r_i^k \in \mathbb{N}\}$; (ii) the estimated processing time needed in order to finish the activity: p_i .

The combination of the resource requirement and the estimated processing time would permit the activity a_i to be finished with the given resources in the given processing time. We call such a combination an *activity operating mode* or simply a *mode*. Let μ_i be a mode of activity a_i , and

$$\mu_i = \langle \{(k: r_i^k) | k \in \{1, \dots, K\} \wedge r_i^k \in \mathbb{N}\}, p_i \rangle. \quad (2.5)$$

³We note that each activity may require more than one resource type for execution. In case an activity requires 0 unit of resource type R_k , the item $(k: 0)$ in the resource requirement set is omitted for brevity purpose. One exception is for the dummy activities, of which the resource requirement of dummy activities are written as $\{(0: 0)\}$.

An activity may sometimes be carried out by using more than one mode. Problems with multiple mode options for executing activities are termed *multi-mode* RCPSP (MRCPSP). MRCPSP are not the subject of this research; for an impression, the readers are referred to the survey work on this subject by Lova et al. (2006).

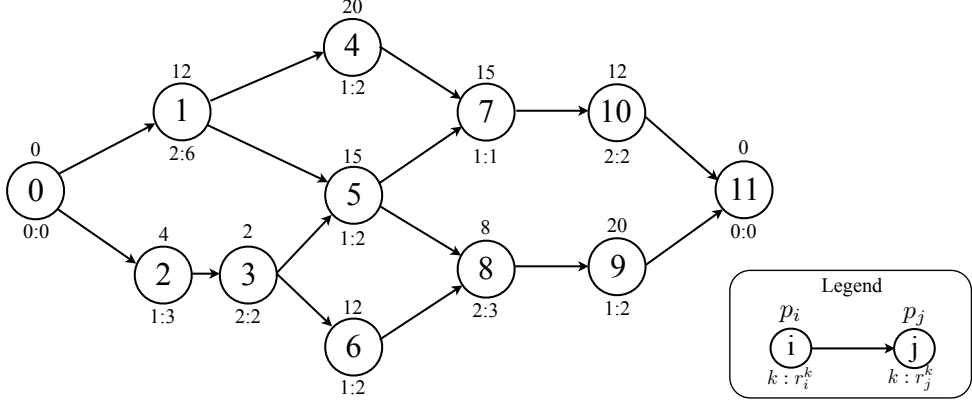


Figure 2.6: Refined AoN representation of a project

The AoN network in Figure 2.6 details the AoN network of the project (as given in Figure 2.1) by associating with each activity a mode μ_i . For simplicity, in the given example, we assume that performing an activity requires only one of the two resource types ($\mathcal{R} = \{R_1, R_2\}$). For instance, the mode of activity a_5 in the project depicted in Figure 2.6 is $\langle \{(1:2)\}, 15 \rangle$, meaning that the execution of activity a_5 requires 2 units of resource type R_1 and lasts 15 time units.

C: Resource-capacity Constraints

Traditionally, resource constraints in scheduling problems refer to the resource-capacity constraint. When resources are capacity constrained, it means that for each resource type R_k ($R_k \in \mathcal{R}$), at most $\bar{c}_k \in \mathbb{N}$ units of the resource type can be used at the same time, where $\bar{c}_k \in \mathbb{N}$ is the maximum capacity of resource type R_k . We recall that r_i^k is the amount of resource type R_k used by activity a_i ($a_i \in A$). We assume that the given quantity \bar{c}_k is constant throughout the scheduling horizon. The same holds for r_i^k throughout the processing duration of a_i .

Given a complete set of activity start times $S = \{s_i\}_{a_i \in A^+}$ of the project, let

$$A(S, t) = \{a_i \in A | s_i \leq t < s_i + p_i\} \quad (t \geq 0) \quad (2.6)$$

be the set of activities of which the processing times contain the time point t , also called the *active set* at time t . Let $u_k(S, t)$ be the amount of resource type R_k used at time t by all activities. So,

$$u_k(S, t) = \sum_{a_i \in A(S, t)} r_i^k \quad (R_k \in \mathcal{R}, t \geq 0). \quad (2.7)$$

Moreover, the resource-capacity constraint can be formulated as

$$u_k(S, t) \leq \bar{c}_k \quad (R_k \in \mathcal{R}, t \geq 0). \quad (2.8)$$

2.1.4 Schedules and Performance Measures

A sequence of scheduled start times $S = (s_0, \dots, s_{n+1})$ for all activities of a project, is called a *project schedule*. A project schedule is a solution to an RCPSP. In this subsection, we first define the concept of *feasibility* of a schedule and then discuss several scheduling objectives, i.e., different ways of measuring the performance of a schedule.

A schedule to an RCPSP is called **feasible** if all precedence constraints and resource constraints are satisfied. We define a feasible schedule as follows.

DEFINITION 2.7 Feasible schedule. *A feasible schedule S to an RCPSP should satisfy the following constraints simultaneously.*

$$\begin{aligned} \overline{rl} &\leq s_0 \\ s_i + p_i &\leq s_j \quad (a_i \prec a_j) \\ u_k(S, t) &\leq \bar{c}_k \quad (R_k \in \mathcal{R}, t \geq 0) \end{aligned} \quad (2.9)$$

A feasible schedule S to a TRCPSP should satisfy an additional constraints: $s_{n+1} \leq \overline{dl}$.

In the scheduling phase of project management, finding a feasible project schedule is essential. However, when a project has more than one feasible schedule, in most cases the project manager will try to find the best schedule amongst all feasible schedules using suitable objective functions. In the following subsection, we discuss various project-scheduling objectives.

The quality of a feasible schedule can be measured by a *utility function* (a.k.a. *objective function*) $f(S)$, which represents a particular scheduling objective. A scheduling problem with a utility function becomes an optimisation problem in which $f(S)$ is to be maximised (or minimised). We describe an RCPSP in the following linear programming formulation.

$$\begin{aligned} \text{Find} \quad & S = \arg \min_S f(S) \\ \text{subject to} \quad & s_i + p_i \leq s_j \quad (a_i \prec a_j) \\ & \overline{rl} \leq s_0 \\ & u_k(S, t) \leq \bar{c}_k \quad (R_k \in \mathcal{R}, t \geq 0) \end{aligned} \quad (2.10)$$

Motivated by real-world situations, a wide variety of objectives for RCPSP have been studied. We distinguish two classes of objectives: (a) *time-based objectives* and (b) *resource-based objectives*. In addition, most real-world applications consider a third class, i.e., (c) a combination of multiple objectives as a joint objective. In the sequel, we will discuss and formulate these three classes of project-scheduling objectives.

A: Time-based Objectives

One of the most common objectives in project scheduling is to find the schedule that minimises the *project makespan* (a.k.a. the *project throughput time*). Project makespan is defined as the elapsed time between the project release and the project completion. Let PM denote the objective of (1) *minimising the project makespan*. We formulate the objective of PM as follows.

$$\text{PM:} \quad f(S) = s_{n+1} - \overline{rl} \quad (2.11)$$

Minimising the project makespan is important in many practical situations: it leads to a timely release of resource capacities for future projects; it reduces the risk of violating a deadline; it generates timely incoming cash flows, etc. (cf. Demeulemeester and Herroelen, 2002).

When a project due time \overline{dt} is given, a representation variation of minimising the project makespan, considered as the second time-based objective, is (2) *minimising the project delay* (denoted by PD). Project delay is often referred to as the elapsed time between the project due time and the project completion time.

$$\text{PD:} \quad f(S) = \max(s_{n+1} - \overline{dt}, 0) \quad (2.12)$$

B: Resource-based Objectives

In many real-world situations resource-based objectives are considered next to time-based objectives. Below, we introduce two often-used resource-based objectives.

If the resources necessary to carry out the activities have to be purchased (e.g., expensive machinery), then we speak of the **resource investment problem**. Project managers in resource investment problems often want to *minimise the total resource procurement cost* (denoted by TRPC). The objective function for minimising the total resource procurement cost is defined as follows.

$$\text{TRPC:} \quad f(S) = \sum_{R_k \in \mathcal{R}} c_k^p \max_{t \geq 0} u_k(S, t), \quad (2.13)$$

where $c_k^p \geq 0$ is the *procurement cost* per unit of resource type $R_k \in \mathcal{R}$ and $u_k(S, t)$ is the amount of resource R_k used at time t given a schedule S .

The second resource-based objective involves generating a schedule where the utilisation of resources is as flat as possible, without violating the project-deadline constraint. In this case, we speak of the **resource levelling problem**. The degree to which the resource usage is levelled can be expressed in various ways (cf. Neumann et al., 2001). A typical resource-levelling measurement considers the *total squared resource utilisation cost*. The objective of *minimising the total squared resource utilisation* (denoted by TSRU) given a schedule S can be formulated as follows.

$$\text{TSRU:} \quad f(S) = \sum_{R_k \in \mathcal{R}} \sum_{t \geq 0} u_k^2(S, t) \quad (2.14)$$

When utilisation resources of different resource types is charged with different cost, we speak of *minimising the total squared resource utilisation cost* (denoted by TSRUC), and it can be formulated as follows.

$$\text{TSRUC:} \quad f(S) = \sum_{R_k \in \mathcal{R}} c_k^u \sum_{t \geq 0} u_k^2(S, t) \quad (2.15)$$

where $c_k^u \geq 0$ is the *utilisation cost* per unit of resource type $R_k \in \mathcal{R}$ per unit of time.

C: A Combination of Multiple Objectives

From the discussion above, we have seen that a project schedule can be weighted against a variety of performance measures. These measures may pertain to the makespan of the project, the delay of the project, the resource procurement, the levelling of resource utilisation, etc. In many situations, these objective functions may be more or less equally relevant. A solution that is optimal with respect to one single objective might be arbitrarily bad with respect to other criteria, and thus unacceptable for a project manager (T'kindt and Billaut, 2006). In general, there will be a trade-off amongst schedules. This forces a project manager to decide a weight distribution for each of the measures in such situations.

This gives rise to the problem of scheduling projects under multiple objectives (cf. Słowiński et al., 1994; T'kindt and Billaut, 2006). The problem is also sometimes referred to as *multi-goal* problem or *multicriteria* problem. The analysis involves the use of different objectives which are combined with *weight factors*. A weight factor that is assigned to each of the considered objectives, determines the importance of one objective vis-à-vis that of other objectives. We note that these weight factors should be context dependent and they need to be empirically modelled.

Below we give an example of a combination of two objectives: minimising the project delay (PD) and minimising the total squared resource utilisation cost (TSRUC).

$$\text{PD + TSRUC:} \quad f(S) = w_1 \max(s_{n+1} - \bar{d}t, 0) + w_2 \sum_{R_k \in \mathcal{R}} c_k^u \sum_{t \geq 0} u_k^2(S, t) \quad (2.16)$$

In Equation 2.16, the two weight factors w_1 and w_2 are used to represent the relative importance of one objective compared to the other.

2.2 AGH Scheduling Problem

The AGH scheduling problem has been brought up only recently, namely since the liberalisation of ground-handling market (cf. Schmidberger et al., 2009). Models such as *job*

shop scheduling (see Xue and Fan, 2007) and *simple temporal network* (see van Leeuwen and Witteveen, 2009) are employed to formulate the problem. In this section, we identify and analyse the characteristics of an AGH scheduling problem from a project-scheduling perspective.

In the research field of project scheduling, the RCPSP addressed in Section 2.1 is a well-known problem. However, it is a rather basic model with assumptions that are too restrictive for many practical applications (Hartmann and Briskorn, 2010). In practice, project-scheduling problems can be of various variations and extensions of the classic RCPSP model⁴.

In this section, we focus on three extensions of classic RCPSP with considering the characteristics of the studied AGH scheduling domain. They are scheduling multiple projects (see §2.2.1), (2) informational and managerial decentralisation (see §2.2.2), and uncertainty in scheduling (see §2.2.3). Following the discussion of the three extensions of RCPSP, we model the AGH scheduling problem as an instance of a generalised RCPSP — a decentralised resource-constrained multi-project scheduling problem under uncertainty (see §2.2.4).

2.2.1 Scheduling Multiple Projects

In real-life applications, it is rarely the case that a single organisation carries out a single project at a time. Instead, an increasing number of organisations tend towards an organisational structure in which multiple projects are performed simultaneously, and with collaboration of a number of partners (cf. Pennypacker and Dye, 2002; Tobis and Tobis, 2002; Turner, 2008).

As introduced in Section 1.1, an aircraft turnaround process consists of a series of ground-handling operations. The operations can be seen as the intermediate activities in the scope of a project. Therefore, scheduling a turnaround process can be seen as an instance of an RCPSP. Consequently, AGH scheduling, which aims at scheduling all aircraft turnarounds at an entire airport, requires an extended model of RCPSP with multiple projects.

In a multi-project scheduling context, assuming that the activities of each project require only local (non-shared) resources, the problem of scheduling multiple projects can be decomposed into scheduling a set of independent (single) projects (Confessore et al., 2007). However, most of the projects carried out in a multi-project environment do not have the luxury of dedicated resources. A number of researchers (e.g., Payne, 1995; Pennypacker and Dye, 2002) explicitly pointed out that most projects that are run in parallel by a company make use of shared and often limited resources. Frequent conflicts of interest arise when more than one project requires the same type of resource at the same time. Therefore, scheduling multiple projects with shared resources is much more complicated than scheduling in single-project cases.

We define the problem of scheduling multiple projects sharing limited resources as a *resource-constrained multi-project scheduling problem*.

⁴While we are composing this thesis, Hartmann and Briskorn (2010) published a survey work on variants and extensions of the classic RCPSP. The survey provides an exhaustive overview over the extensions studied in the last decades (restricted to deterministic problems though), and classifies the extensions according to the structure of the RCPSP.

DEFINITION 2.8 Resource-constrained Multi-project Scheduling Problem

(RCMPSP). A resource-constrained multi-project scheduling problem is the problem of simultaneous scheduling two or more projects, each of which has its own set of activities constituting different network structures. A common pool of resources is provided to execute the activities of different projects. Precedence relations between two activities are defined only within the same project.

Formally, we can describe an RCMPSP as follows.

- An RCMPSP consists of a set \mathcal{P} of m projects ($\mathcal{P} = \{P_1, \dots, P_m\}$, $m \in \mathbb{N}_{\geq 2}$), sharing a set \mathcal{R} of K types of (renewable) resources ($\mathcal{R} = \{R_1, \dots, R_K\}$, $K \in \mathbb{N}_{\geq 1}$).
- Each project P_i ($i \in \{1, \dots, m\}$) has a release time \overline{r}_i , and consists of a set A_i of $n_i \in \mathbb{N}$ real activities $a_{i,j}$ with $j \in \{1, \dots, n_i\}$. Two fictitious activities $a_{i,0}$ and a_{i,n_i+1} are added for representing the *start* and the *completion* of project P_i .
- Each resource type R_k has a maximum capacity \overline{c}_k .
- Each activity $a_{i,j}$ has only one activity operating mode $\mu_{i,j} = \langle \{(k: r_{i,j}^k) \mid k \in \{1, \dots, K\} \wedge r_{i,j}^k \in \mathbb{N}\}, p_{i,j} \rangle$
- Simple finish-start precedence relations \prec describe execution orders for pairs of activities of the same project: $a_{i,j} \prec a_{i,l}$.

A feasible solution to an RCMPSP is a schedule $S = \{s_{i,j} \mid 1 \leq i \leq m, 0 \leq j \leq n_i + 1\}$ specifying the start times of all project activities and satisfying the following constraints.

$$\begin{aligned}
 \overline{r}_i &\leq s_{i,0} & (i \in \{1, \dots, m\}) \\
 s_{i,j} + p_{i,j} &\leq s_{i,l} & (a_{i,j} \prec a_{i,l}) \\
 u_k(S, t) &\leq \overline{c}_k & (R_k \in \mathcal{R}, t \geq 0)
 \end{aligned} \tag{2.17}$$

We recall that function $u_k(S, t)$ stands for the amount of resource type R_k used at time t by all project activities $\{a_{i,j}\}$. It can be derived as follows.

$$u_k(S, t) = \sum_i \sum_j r_{i,j}^k, \text{ where } s_{i,j} \leq t < s_{i,j} + p_{i,j} \tag{2.18}$$

A variety of objective functions differing from those for RCPSP have to be considered when solving an RCMPSP. We mention in the following several widely used examples of these objective functions: (1) TPM: *minimising the total project makespan*, (2) SPM: *minimising the summed project makespan*, and (3) SPD: *minimising the summed project delay*, (4) APD: *minimising the average project delay*, (5) APDP: *minimising the average project delay penalty*.

TPM is an objective function on the top management level regardless of the relative importance of different projects. It concerns the time difference between the completion time of the last project and the release time of the first project. The formulation of TPM is as follows.

$$\text{TPM:} \quad f(S) = \max_i(s_{n_i+1}) - \min_i(\overline{r}l_i), \quad \forall i \in \{1, \dots, m\} \quad (2.19)$$

The objective function of SPM is defined as follows.

$$\text{SPM:} \quad f(S) = \sum_{i=1}^m (s_{i,n_i+1} - \overline{r}l_i) \quad (2.20)$$

Assuming due time \overline{dt}_i of each project P_i is introduced, we define the objective function of SPD as follows.

$$\text{SPD:} \quad f(S) = \sum_{i=1}^m \max(s_{i,n_i+1} - \overline{dt}_i, 0) \quad (2.21)$$

We recall that project makespan is defined as the elapsed time between project release time and project completion (see Equation 2.11). Accordingly, we have the objective function of APD as follows.

$$\text{APD:} \quad f(S) = \sum_{i=1}^m \frac{\max(s_{i,n_i+1} - \overline{dt}_i, 0)}{m} \quad (2.22)$$

The APD objective less accurately represents the reality, as it implicitly assumes equal delay penalties for all projects (cf. Kurtulus, 1985). APDP is introduced as a more realistic objective, where each project is associated with a time-unit delay cost (i.e., c_i^{dl} for P_i).

$$\text{APDP:} \quad f(S) = \sum_{i=1}^m \frac{c_i^{dl} \cdot \max(s_{i,n_i+1} - \overline{dt}_i, 0)}{m} \quad (2.23)$$

2.2.2 Informational and Managerial Decentralisation

In the AGH environment, having a centralised authority that makes the scheduling decisions for all aircraft on each individual ground-handling operation is impractical and undesirable. First, aggregating all aircraft-turnaround information to a central hub is a heavy task with respect to both computation and communication. Second, aircraft are operated by different airline companies that may have different interests and preferences. Third, ground-handling operations are carried out by a number of self-interested third-party ground-service providers.

The informational and managerial decentralisation in AGH scheduling problem coincide with many real-world project-scheduling situations where project environments are becoming more distributed both geographically and organisationally (cf. Confessore et al.,

2007). In this case, the multiple projects in an RCMPSP usually will no longer belong to the same company. Each project may have its own manager, who is self-interested and tries to optimise the performance of its own project. Information about the precedence relations among activities as well as the resource requirement of one project is only known to the project manager himself⁵. Moreover, resources shared by the multiple projects in an RCMPSP may be provided by various resource providers. We assume that each resource type is associated with a resource provider who is also self-interested and tries to maximise its own utility function with regard to the utilisation of the resources it provides.

The decentralisation of decision-making processes in practice urges us to study a decentralised scheduling problem. We define a *decentralised resource-constrained multi-project scheduling problem* as follows.

DEFINITION 2.9 Decentralised Resource-constrained Multi-project Scheduling Problem (DRCMPSP). *A decentralised resource-constrained multi-project scheduling problem is an RCMPSP in which each project is managed by a self-interested project manager, and each resource type is managed by a self-interested resource manager. All different types of managers make scheduling decisions based on their own objectives.*

We note that the DRCMPSP definition given in Definition 2.9 extends the decentralised problem definition introduced by Confessore et al. (2007) and Homberger (2007) by introducing the self-interested resource managers.

2.2.3 Decision Making under Uncertainty

Uncertainty is inevitable in real-life project environment. Project-management uncertainty may have a variety of sources. These include imprecise, outdated or incomplete information, inability to accurately model the impact of expected or unexpected events, imprecision in estimation and judgement, and so on.

An attempt to categorise uncertainty in project management was made by Bonfill-Teixidor (2006) from a hierarchical decision-making process' point of view. She divided uncertainty into three categories: *strategic uncertainty*, *tactical uncertainty*, and *operational uncertainty*. Project scheduling involves decisions on the operational level of project management. Therefore, we focus on studying the operational uncertainty. Below we give several examples of uncertainty⁶ with respect to projects and activities, respectively.

- **Uncertainty with respect to projects**

1. A project may have a delayed or advanced release time.
2. A project deadline may be postponed or advanced.
3. New arriving project(s) may have to be incorporated.
4. Certain projects may be cancelled before the projects start or during the course of the project executions.

⁵For brevity, we use 'he' or 'his' whenever 'he or she' and 'his or her' are meant.

⁶For the sake of readability, we abbreviate the term "operational uncertainty" to "uncertainty" here and for the rest of the thesis, too.

- **Uncertainty with respect to activities**

1. The processing time and resource requirement of an activity may be inaccurately estimated.
2. Resources for carrying out an activity may become unavailable or arrive behind schedule.
3. Staff/operators may encounter problems that cause productivity drops.
4. New activities may have to be incorporated during the course of a project.
5. Certain activities may have to be dropped due to changes in the project scope.

As it is clear from the given examples, uncertainty lies at the very heart of project-scheduling problems. Below, we define a DRCMPSP under uncertainty.

DEFINITION 2.10 Decentralised Resource-constrained Multi-project Scheduling Problem under Uncertainty (DRCMPSP/u). *A decentralised multi-project scheduling problem under uncertainty is a DRCMPSP in which projects' executions are subject to various uncertainty.*

Similar to most practical project-scheduling environments, the AGH scheduling environment is well known for its large number of disturbances stemming from various sources. The disturbances cause a high degree of uncertainty in the AGH schedules. In this thesis, we focus on investigating scheduling solutions under the following two classes of uncertainty: (a) *partial observability* and (b) *nondeterminism*. Below, we discuss them briefly in the context of AGH scheduling.

A: Partial Observability

Partial observability in AGH scheduling can be interpreted as variable aircraft arrival times. The actual arrival time of an aircraft at the airport is often different from (most of the times, it is later than) the one foreseen in the original flight plan. The uncertainty in aircraft arrival time may be caused by different reasons, for instance, (i) bad weather conditions at the arrival airport for aircraft landing, (ii) the delayed departure from the departure airport, and (iii) longer en-route flying time because of head wind or air traffic control. In project-scheduling terms, partial observability is interpreted as variable project release times⁷.

⁷A recent work of investigating decentralised decision making in partial observable environments can be found in Oliehoek (2010), in which the author investigated the *decentralised partially observable Markov decision process* (Dec-POMDP). The research scope and focus of Oliehoek (2010) are different from ours in three distinct aspects. First, Oliehoek (2010) studies *planning* problem, whereas we study *scheduling* problem. Second, Oliehoek (2010) studies *cooperative* agents, whereas we study *competitive* agents. Third, in Oliehoek (2010) the uncertainty resides in both the outcome of agent actions and the perception of the state of the environment, whereas in our research we consider the uncertainty in the perception of the state of the environment only.

B: Nondeterminism

Nondeterminism in AGH scheduling can be interpreted as variable ground-handling operational times. During the scheduling phase of AGH, each of the ground-handling operations has an estimated processing duration. However, during the execution phase, the actual processing durations may differ from the original estimations. There can be various reasons for this variation, for instance, (i) bad weather conditions will extend the baggage loading/unloading duration, (ii) breakdown of maintenance machinery will require extra time for aircraft maintenance, and (iii) no-show of passengers will delay the passenger boarding process. In project-scheduling terms, nondeterminism is interpreted as variable activity processing times.

2.2.4 A DRCMPSP/u Formulation of AGH Scheduling Problem

Based on the analyses above, we can formulate the AGH Scheduling problem as an instance of a DRCMPSP/u. A formal definition of an AGH scheduling problem is given below.

DEFINITION 2.11 AGH Scheduling Problem. *An AGH scheduling problem is the problem of scheduling a set \mathcal{P} of m aircraft turnaround processes ($\mathcal{P} = \{P_1, \dots, P_m\}$, $m \in \mathbb{N}_{\geq 2}$), where*

- P_i ($i \in \{1, \dots, m\}$) consists of a set A_i of $n_i \in \mathbb{N}$ ground-handling operations $a_{i,j}$ with $j \in \{1, \dots, n_i\}$;
- Two fictitious operations $a_{i,0}$ and a_{i,n_i+1} are added for representing the start and the completion of P_i ;
- Each aircraft is managed by a manager who makes scheduling decisions for the aircraft's ground-handling operations;
- Each aircraft has an expected arrival time \overline{rl}_i and an expected departure time \overline{dt}_i according to the flight timetable;
- All aircraft share a set \mathcal{R} of K types of (renewable) resources ($\mathcal{R} = \{R_1, \dots, R_K\}$, $K \in \mathbb{N}^+$), each of which is provided by a self-interested ground-service provider;
- Operation $a_{i,j}$ has only one operating mode $\mu_{i,j} = \langle \{(k: r_{i,j}^k) \mid k \in \{1, \dots, K\} \wedge r_{i,j}^k \in \mathbb{N}\}, p_{i,j} \rangle$;
- Simple finish-start precedence relations \prec describe execution orders for pairs of operations in P_i : $a_{i,j} \prec a_{i,l}$, and it is only known to the manager of P_i ;
- The actual arrival time \overline{rl}_i^* of an aircraft P_i may be different from the expected one;
- The actual processing time $p_{i,j}^*$ of a ground-handling operation $a_{i,j}$ may be different from the estimated one;

The last two items in Definition 2.11 represent partial observability and nondeterminism, respectively.

2.3 Chapter Summary

In this chapter we formulated the AGH scheduling problem into a well-studied scheduling-problem framework — the project scheduling problem. We provided the definition of the classic project scheduling problem — RCPSP, as well as some extensions of RCPSP.

The RCMPSP as a first extension of RCPSP is defined as a problem in which two or more projects are concurrently active at a point in time. Each project, usually represented by means of a network, contains a finite set of activities which are ordered by precedence relations. These projects can have different and often conflicting properties, such as different release times, different urgencies represented by deadlines, and different objectives.

The DRCMPSP further extends the RCMPSP by addressing multiple self-interested decision makers. In a decentralised setting, not only the project managers, but also the resource-type managers make strategic decisions to fulfil their own goals.

Moreover, we investigate the DRCMPSP with the existence of uncertainty, making it a DRCMPSP/u. We identify two classes of uncertainty — partial observability and nondeterminism — that appear in AGH scheduling problems as variable aircraft arrival times and variable ground-handling operational times, respectively.

Finally, the AGH scheduling problem is formulated as an instance of a DRCMPSP/u. Different aircraft managers being project managers have different and independent objectives, and ground-service providers being resource managers of different resource types also have their own objective with respect to the utilisation of their resources.

Chapter 3

A Review of Existing Solution Methods

The field of project-management theory and practice has taken tremendous strides forward in the past few decades (Demeulemeester and Herroelen, 2002). Over the years, the project-scheduling problems occurring in practice exhibit more and more complex structure, evolving from scheduling a *static and deterministic small-scale single* project to scheduling *dynamic and nondeterministic large-scale multiple* projects. The development of solution procedures over the years for their resolution transcend the project management area.

In this chapter, we provide a literature review on the latest development of project-scheduling research with emphasis on the problems where (1) multiple projects have to be scheduled using shared resources, (2) information about projects and resources is asymmetric and decision-making processes are decentralised (3) projects are executed in an environment with various uncertainties.

We start this chapter by investigating the existing approaches to RCMPSPs (see Section 3.1). Subsequently, we provide an extensive survey on the latest development of solution methods to DRCMPSPs (see Section 3.2). Furthermore, we review some of the methods that have been proposed in the literature for scheduling under uncertainty (see Section 3.3). Section 3.4 summarises this chapter by discussing to which extent we go beyond state of the art concerning the solution methods and thus contribute to the field.

3.1 Solution Methods for RCMPSP

In the past, various methods from both operations research and artificial intelligence were proposed to handle RCMPSP in a centralised manner (cf. Lova and Tormos, 2001; Hans et al., 2007). In the case all projects were managed by a single authority, a centralised scheduling method would be applied to find an integrated schedule. Two classes of approaches have been used in centralised scheduling (cf. Lova and Tormos, 2001): (a) *single-project approaches* and (b) *multi-project approaches*. Below, we briefly discuss the

difference between these two classes of approaches and focus on investigating solution methods using multi-project approaches.

A: Single-project Approaches

In a single-project approach, the individual projects of an RCMPSP is aggregated into a single mega-project, which can then be represented by a super AoN network (see Figure 3.1). The super AoN network combines all projects' activity networks and adds two super-dummy activities — a super-dummy start activity (node “0” in Figure 3.1) and a super-dummy completion activity (node “ $n + 1$ ” in Figure 3.1). Both of the two super-dummy activities are associated with mode $\langle \{(0: 0)\}, 0 \rangle$.

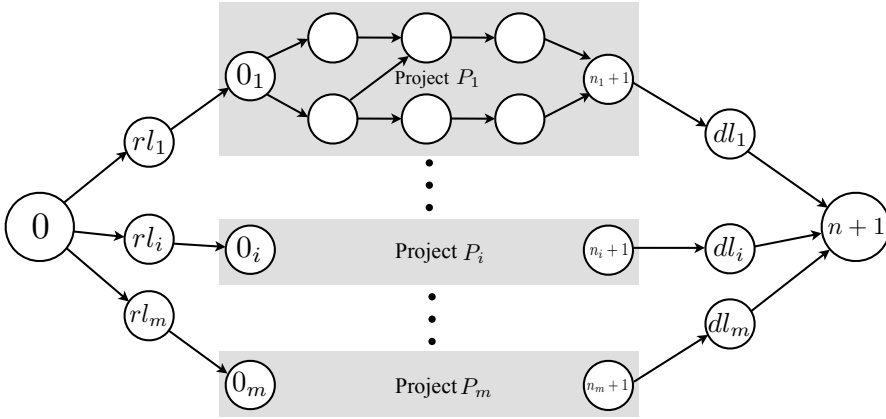


Figure 3.1: An example of super AoN network for RCMPSP

When release-time constraints exist, a super release time \overline{Rl} of the mega-project is defined as the earliest release time among all project release times, and

$$\overline{Rl} = \min_i(\overline{rl}_i), \quad \forall i \in \{1, \dots, m\}.$$

Similarly, when project-deadline constraints exist, a super deadline \overline{Dl} for the mega-project is defined as the latest deadline among all project deadlines, and

$$\overline{Dl} = \max_i(\overline{dl}_i), \quad \forall i \in \{1, \dots, m\}.$$

In a super AoN network, the release time and the deadline of an individual project P_i can be imposed by inserting two additional fictitious activities, $a_{i,rl}$ and $a_{i,dl}$, where

$$\begin{aligned} p_{i,rl} &= \overline{rl}_i - \overline{Rl}, \\ p_{i,dl} &= \overline{Dl} - \overline{dl}_i. \end{aligned} \tag{3.1}$$

Therefore, the mode of dummy activity $a_{i,rl}$ is $\langle \{(0: 0)\}, p_{i,rl} \rangle$, and the mode of dummy activity $a_{i,dl}$ is $\langle \{(0: 0)\}, p_{i,dl} \rangle$.

Aggregating multiple projects into a mega-project reduces an RCMPSP to an RCPSP, which can be subsequently solved by an RCPSP solver. For the literature of solving RCPSP, readers are referred to the survey work by Özdamar and Ulusoy (1995) and Herroelen et al. (1998), as well as to two handbooks by Neumann et al. (2001) and Demeulemeester and Herroelen (2002).

Employing a single-project approach for solving RCMPSP is not always applicable. First, aggregating multiple projects into a mega-project implicitly assumes equal delay penalties for all projects, which is rarely the case in practice. Second, independent project analysis becomes difficult when all projects are bound together. Third, single-project approaches cannot address the different importance of the projects in an RCMPSP. Fourth, aggregating multiple projects into a mega-project significantly increases the network complexity and consequently increases the computational effort of the solution procedures.

Therefore, solution methods that can solve an RCMPSP while maintaining a separate critical path per project are needed. These solution methods are often referred to as *multi-project approaches* or *parallel scheduling approaches* (cf. Lova and Tormos, 2001; Krüger and Scholl, 2007).

B: Multi-project Approaches

Solution methods for RCMPSPs based on the multi-project approach fall into four categories: (1) exact methods, (2) priority-rule-based heuristics, (3) meta-heuristics, and (4) constraint satisfaction and optimisation.

3.1.1 Exact Methods

Exact methods aim at finding the optimal solution to a problem. Therefore, they are often referred to as optimal procedures. Optimal procedures for solving RCMPSP have been proposed since the early days of project management research. Examples of such procedures include *zero-one linear programming* (e.g., Pritsker et al., 1969; Deckro et al., 1991), and the *branch-and-bound* algorithm (e.g., Vercellis, 1994).

The pioneering work on multi-project scheduling by Pritsker et al. (1969) proposed a zero-one linear programming formulation for RCMPSP. A three-project, eight-activity (in total), three-resource-type example RCMPSP is used to test the optimal solutions obtained by the zero-one programming code with respect to solutions obtained by two heuristic approaches¹: *first come first served*, and *minimum project slack time first*. They compared the scheduling results with respect to three different scheduling objectives: SPM, TPM, and SPD (see §2.2.1).

Deckro et al. (1991) have formulated RCMPSP as a block angular general integer programming model, and employed a decomposition approach to solve such integer programming problems. The scheduling problem instance they used (which is a problem consisting of eight projects) is slightly larger and more complex than the one used by Pritsker et al. (1969). However, the solution scalability is still limited.

Vercellis (1994) described a Lagrangian decomposition technique for solving multi-mode RCPSP. The decomposition can be useful in several ways, such as providing bounds

¹A more detailed discussion on heuristic approaches can be found in the §3.1.2.

on the optimum so that the quality of approximate solutions can be evaluated. Furthermore, in the context of branch-and-bound algorithms, it can be used for more effective fathoming of the tree nodes. In the modelling perspective, the Lagrangian optimal multipliers can provide insight to the prices for assigning the resources to different projects.

The RCMPSP is a generalisation of the RCPSP with multiple projects. It has been shown by Błażewicz et al. (1983) that the RCPSP, as a generalisation of the classical job shop scheduling problem, belongs to the class of NP-hard optimisation problems. The RCMPSP problem, as a generalisation of the RCPSP, is therefore also NP-hard, meaning that there are no known algorithms for finding optimal solutions in polynomial time (cf. Garey and Johnson, 1979). Complexity analysis for the RCMPSP is not encouraging, especially because the actual project scheduling problems in real-world applications are often of large-scale. Problems become intractable when using exact methods. Hence, most research has sought (1) efficient heuristics, and (2) meta-heuristics. In the following two subsections, we will discuss the state of the art on priority-rule-based heuristics and meta-heuristics for RCMPSP.

3.1.2 Priority-rule-based Heuristics

Most of the heuristic methods used for solving RCMPSP belong to the class of *priority-rule-based methods* (see Kurtulus and Davis, 1982; Kurtulus and Narula, 1985; Lova and Tormos, 2001; Browning and Yassine, 2010). In general, priority-rule-based heuristics assign different priority values to the activities that request the same resource at the same time. The activity with the highest priority value will make use of the resource first. These methods are also sometimes referred to as *X-pass methods*, including *single-pass methods* and *multi-pass methods*. The difference between single-pass and multi-pass methods is the number of schedules they generate for each problem. Single-pass methods generate a single schedule; while multi-pass methods generate more than one schedule and select the one that optimises the objective function (cf. Kolisch and Hartmann, 1999). Examples of multi-pass methods are: (a) *forward-backward scheduling methods* (cf. Lova et al., 2000) and (b) *sampling methods* (cf. Lova et al., 2000; Lova and Tormos, 2002).

Priority rules can be classified on the basis of the information they use: (1) *activity-related*, (2) *project-related*, and (3) *resource-related* (Kolisch and Hartmann, 1999).

- Activity-related rules assign a priority value to an activity based on a parameter or characteristic of the activity itself, such as its processing time (e.g., shortest processing time first) or slack time (e.g., minimum slack time first).
- Project-related rules assign priorities to activities based on the project they belong to, or characteristics of that project (e.g., shortest activity from shortest project first).
- Resource-related rules assign priority in terms of an activity's resource demands, scarcity of resources used, or some combination. High priorities are usually assigned to potential bottleneck activities. An example is the maximum-total-work-content rule).

Some rules combine elements of information about the activity, the project, and the resources (cf. Kolisch and Hartmann, 1999).

Kurtulus and Davis (1982) developed six priority rules for the multi-project environment, and along with three single-project priority rules. They analysed these rules with the objective of SPD; they found that shortest-processing-time-first heuristic was the best under most conditions. Browning and Yassine (2010) have extended the work by Kurtulus and Davis (1982) and conducted experiments with 20 priority rules on in total 12,320 test problems. We summarise twenty priority rules mainly studied by Kurtulus and Davis (1982) and Browning and Yassine (2010) in Table 3.1.

Table 3.1: Priority-rule-based heuristics

	Priority Rules	Basis	Description and Formula	References
	FCFS -			
1.	First Come First Served	Activity	$\min(t_{i,j}^{es})$, where $t_{i,j}^{es}$ is the earliest start time of $a_{i,j}$	Kurtulus and Davis (1982); Lova and Tormos (2001)
	LCFS -			
2.	Last Come First Served	Activity	$\max(t_{i,j}^{es})$	Browning and Yassine (2010)
3.	RAN - Random	Activity	Activities selected randomly	Browning and Yassine (2010)
	SPT -			
4.	Shortest Processing Time	Activity	$\min(p_{i,j})$ where $p_{i,j}$ is the processing time of $a_{i,j}$	Kurtulus and Davis (1982)
	LPT -			
5.	Longest Processing Time	Activity	$\max(p_{i,j})$	Kurtulus and Davis (1982)
6.	MINSLK - Minimum Slack time	Activity	$\min(t_{i,j}^{sl})$ where $t_{i,j}^{sl} = t_{i,j}^{ls} - t_{i,j}^{es}$ is the slack time of $a_{i,j}$	Fendley (1968); Lova and Tormos (2001); Kurtulus and Davis (1982)
7.	MAXSLK - Maximum Slack time	Activity	$\max(t_{i,j}^{sl})$	Kurtulus and Davis (1982)
8.	SASP - Shortest Activity from the Shortest Project	Activity, Project	$\min(\rho_i + p_{i,j})$, where ρ_i is the length of critical path of project P_i	Kurtulus and Davis (1982); Deckro et al. (1991); Lova and Tormos (2001)
9.	LALP - Longest Activity from the Longest Project	Activity, Project	$\max(\rho_i + p_{i,j})$	Kurtulus and Davis (1982)
10.	MINTWK - Minimum Total Work content	Activity, Resource	$\min \left(\sum_{k=1}^K \sum_{j \in A_i^s} p_{i,j} r_{i,j}^k + p_{i,j} \sum_{k=1}^K r_{i,j}^k \right)$, where A_i^s is the set of activities already scheduled in project P_i	Kurtulus and Davis (1982); Lova and Tormos (2001)

Table 3.1 – continued from previous page

	Priority Rules	Basis	Description and Formula	References
11.	MAXTWK - Maximum Total Work content	Activity, Resource	$\max \left(\sum_{k=1}^K \sum_{j \in AS_i} p_{i,j} r_{i,j}^k + p_{i,j} \sum_{k=1}^K r_{i,j}^k \right)$	Kurtulus and Davis (1982); Lova and Tormos (2001)
12.	MINLST - Minimum Latest Start Time	Activity	$\min(t_{i,j}^{ls})$	Browning and Yassine (2010)
13.	MINLFT - Minimum Latest Finish Time	Activity	$\min(t_{i,j}^{lf})$	Lova and Tormos (2001); Browning and Yassine (2010)
14.	MAXSP - Maximum Schedule Pressure	Activity	$\max \left(\frac{t - t_{i,j}^{lf}}{p_{i,j} W_{i,j}} \right)$, where $W_{i,j}$ is the percentage of the activity remaining to be done at time t	Browning and Yassine (2010)
15.	MINWCS - Minimum Worst Case Slack	Activity, Resource	$\min(t_{i,j}^{ls} - \max(E_{i,j}^{i',j'} (a_{i,j}, a_{i',j'}) \in A_t^p))$, where $E_{i,j}^{i',j'}$ is the earliest time to schedule activity $a_{i',j'}$ if $a_{i,j}$ is started at time t , and A_t^p is the set of all feasible pairs of eligible, un-started activities at time t	Browning and Yassine (2010)
16.	WACRU - Weighted Activity Criticality and Resource Utilisation	Activity, Resource	$\max(wCt(a_{i,j}) + (1-w) \sum_{k=1}^K \frac{r_{i,j}^k}{R^k})$, where $Ct(a_{i,j})$ is the criticality of activity $a_{i,j}$	Browning and Yassine (2010)
17.	TWK-LST - Maximum Total Work content & earliest Latest Start Time first (2-phase rule)	Activity, Resource	MAXTWK first, tie broken by $t_{i,j}^{ls}$	Lova and Tormos (2001); Browning and Yassine (2010)
18.	TWK-EST - Maximum Total Work content & earliest Earliest Start Time first (2-phase rule)	Activity, Resource	MAXTWK first, tie broken by $t_{i,j}^{es}$	Browning and Yassine (2010)
19.	MTS - Maximum Total Successors	Activity	$\max(\vec{A}_{i,j}^*)$, where $ \vec{A}_{i,j}^* $ is the total number of transitive successors of $a_{i,j}$	Browning and Yassine (2010)
20.	MCS - Maximum Critical Successors	Activity	$\max(\vec{A}_{i,j}^c)$, where $\vec{A}_{i,j}^c$ is the set of critical successors of $a_{i,j}$	Browning and Yassine (2010)

It is important to note that single-project approaches and multi-project approaches often produce different schedules with the same priority rule (Lova and Tormos, 2001), especially if the rule depends on the critical path (e.g., minimum slack time first). While the single-project approach is more efficient for minimising a single project's processing time, priority rules based on the multi-project approach perform better when minimising the average delay in several projects (cf. Kurtulus and Davis, 1982).

RCMPSP studies disagree about which rule performs best and under which conditions. This disagreement is mainly caused by the fact that problem instances that are used to test the heuristic performance have different characteristics. The recent empirical study conducted by Browning and Yassine (2010) provides a comprehensive analysis on the performance of various priority rules subject to various project-, activity-, resource-related problem characteristics. They have developed a decision table to guide project managers in choosing among the best priority rules based on (1) network complexity and (2) resource-requirement characteristics.

3.1.3 Meta-heuristics

Since the last decade, meta-heuristic solution methods for RCMPSP such as (1) *genetic algorithms*, (2) *simulated annealing*, and (3) *swarm intelligence* started to draw research attention. In general, meta-heuristic methods are used to improve the (preliminary) schedules obtained by priority-rule-based heuristics. We briefly discuss the three best-known meta-heuristics for RCMPSP below.

Genetic Algorithms

Genetic algorithms, inspired by the process of biological evolution, have been introduced by Holland (1975). In contrast to local search strategies, a genetic algorithm simultaneously considers a set or population of solutions instead of only one. Having generated an initial population, new solutions are produced by combining two existing ones (crossover) and/or by altering an existing one (mutation). After producing new solutions, the fittest solutions “survive” and make up the next generation while the others are deleted. The fitness value measures the quality of a solution, usually based on the objective function value of the optimisation problem to be solved. RCMPSP solvers based on genetic algorithms are described by Kim et al. (2005), Kumanan et al. (2006), Yassine et al. (2007), and Gonçalves et al. (2008).

Simulated Annealing

Simulated annealing, introduced by Kirkpatrick et al. (1983), originates from the physical annealing process in which a melted solid is cooled down to a low-energy state. Starting with some initial solution, a so-called neighbour solution is generated by slightly perturbing the current one. If this new solution is better than the current one, it is accepted, and the search proceeds from this new solution. Otherwise, if it is worse, the new solution is only accepted with a probability that depends on the magnitude of the deterioration as well as on a parameter called temperature. As the algorithm proceeds, this temperature is reduced in order to lower the probability to accept worse neighbours. Clearly, simulated

annealing can be viewed as an extension of a simple greedy procedure, sometimes called *first fit strategy*, which immediately accepts a better neighbour solution but rejects any deterioration. RCMPSP solvers based on simulated annealing are described by Shankar and Nagi (1996) and Bouleimen and Lecocq (2000).

Swarm Intelligence

Swarm intelligence is an AI approach based on the collective behaviour of decentralised, self-organised systems. The expression was first introduced by Beni and Wang (1989), in the context of cellular robotic systems. Swarm intelligence is typically made up of a population of simple agents interacting locally with one another and with their environment. The agents follow simple rules, and although there is no centralised control structure dictating how individual agents should behave, local interactions (which are to a certain degree random) between such agents lead to the emergence of “intelligent” global behaviour, mostly unknown to the individual agents. Natural examples of swarm intelligence include ant colonies, bird flocking, animal herding, bacterial growth, and fish schooling. The application of swarm principles to robots is called swarm robotics, while ‘swarm intelligence’ refers to the more general set of algorithms. Deng and Lin (2007) and Gonsalves et al. (2008) have developed two particle swarm optimisation methods to solve RCMPSP.

3.1.4 Constraint Satisfaction and Optimisation

The *constraint satisfaction problem* (CSP) (Kumar, 1992; Tsang, 1993) is a general framework for problems that requires finding states or objects that satisfy a number of constraints or criteria. Here we see that scheduling problems in general are concerned with finding feasible schedules with respect to temporal and/or resource constraints. Therefore, they can be modelled in a CSP framework and solved by CSP solving techniques (Lorterapong and Rattanadamrongagsorn, 2001).

A CSP may have multiple solutions. Any solution to a CSP that models a scheduling problem provides a feasible schedule. In scheduling problems with desired objectives, some solutions are often better than others. In these cases, the tasks of scheduling are to find optimal (or near-optimal) solutions. By adding optimisation criteria, we can model the scheduling problems into a so-called *constraint optimisation problem* (COP) framework whereby the scheduling objective functions are included into a set of constraints.

The generality of the CSP has motivated the development of the *constraint programming* languages and related software systems. These constraint-based systems offer built-in functions for describing common types of constraints and often include CSP-solving techniques developed in CSP research (Ran, 2003). Several constraint programming systems include extensions specifically designed for scheduling applications, e.g., ILOG Scheduler (Nuijten, 1994; Nuijten and Le Pape, 1998) and CHIP (Aggoun and Beldiceanu, 1993). More recent CSP/COP-based solutions for RCPSP are found in the work by Cesta et al. (2002) and Dorndorf (2002).

3.1.5 Beyond Centralised Solution Methods

Nowadays, centralised scheduling approaches to RCMPSPs are losing value due to the fact that projects and resources are managed by independent managers (see §2.2.2). These managers often have different and sometimes conflicting interests and objectives. However, the managers may be independent, the local decision-making processes are not. A coordination mechanism among them is necessary to solve possible conflicts and to allocate the resources in an appropriate way. In Section 3.2, we investigate the solution methods for solving a class of RCMPSP where both the problem information and the decision process are decentralised (i.e., DRCMPSP).

3.2 Solution Methods for DRCMPSP

In decentralised scheduling procedures, scheduling decisions are performed by autonomous decision makers, such as individual resource managers and project managers, taking into account asymmetric information. Information asymmetry is assumed to mean that the problem data of an individual resource type or of an individual project is private. For a resource type, the information privacy means that resource capacity of the resource type, resource allocation over time of the resource type are only known to the resource manager who make scheduling decisions about the resource type. For a project, the information privacy means that the activity network of the project, the processing times of all activities of the project, and information with regard to calculated schedules for the project are only known to the project manager, who makes the scheduling decisions for the project.

The decentralised decision-making process calls for models and techniques that take into account the strategic behaviour of individual decision makers. Therefore, a *multi-agent system*, which can be used to solve problems that are difficult or impossible for an individual agent or monolithic system to solve, is a suitable means for modelling a solution framework to DRCMPSPs. In the following, we provide a brief introduction to multiagent systems and mechanism design (§3.2.1). Four recent agent-based solution methods for solving the DRCMPSP are discussed in §3.2.2.

3.2.1 Multiagent Systems and Mechanism Design

The modern approach to AI is centred around the concept of an *agent*. An agent is anything that can perceive its environment through sensors and act upon that environment through actuators. Such a notion of an agent is fairly general and can include human agents (having eyes/ears/etc. as sensors, hands/legs/etc. as actuators), robotic agents (having cameras as sensors, wheels as actuators), and software agents (having a graphical user interface as sensor and as actuator). From this perspective, AI can be regarded as the study of the principles and the design of artificial agents (Russell and Norvig, 2003).

Three key features are identified while designing an artificial agent. They are *autonomy*, *intelligence*, and *interaction*.

- An agent being **autonomous** means that it is capable of independent action on behalf of its user or owner. In other words, an agent can figure out for itself what

it needs to do in order to satisfy its designed objectives, rather than having to be told explicitly what to do at any given moment.

- **Intelligence** indicates that an agent pursues its goal(s) and executes its tasks in such a way that it tries to optimise some given performance measures. An intelligent agent is sometimes also referred to as a *rational* agent.
- Agents are seldom stand-alone systems. In many situations they coexist and **interact** with other agents in several different ways. Interaction can take place both directly (through a shared language and a communication protocol) and indirectly (through the environment in which they are embedded).

A system that consists of a group of agents that have the potential to interact with each other is called a multiagent system (MAS) (cf. Weiss, 2000; Shoham and Leyton-Brown, 2009; Wooldridge, 2009). In a MAS, agents are assumed to act rationally on behalf of their own interests, and it is generally assumed that their selfish behaviour results in a situation that can be characterised by some sort of system equilibrium (cf. Heydenreich et al., 2006). From a global perspective, such an equilibrium may of course lead to suboptimal system performance.

The following two issues arise in such MAS settings.

- Given a fixed decentralised setting in which agents selfishly act on behalf of their own interests, try to characterise and analyse the quality of the resulting system equilibria from the perspective of the overall system performance.
- Try to design the decentralised setting in such a way that selfish agents are encouraged to show behaviour that results in system equilibria that nevertheless exhibit a good overall system performance.

In economics and game theory, *mechanism design* is the study of designing rules of a game or system to achieve a specific outcome, even though each agent may be self-interested. This is done by setting up a structure in which agents have an incentive to behave according to the rules. The resulting mechanism is then said to implement the desired outcome. The solution concept is related to meta-game analysis, which uses the techniques of game theory to develop rules for a game (cf. Shoham and Leyton-Brown, 2009).

3.2.2 MAS Solutions to DRCMPSP

There is a variety of suitable methods for solving decentralised manufacturing scheduling problems (see Wellman et al. (2001) and Heydenreich et al. (2006)). However, in contrast to the multi-project problem considered here, which is based on a complex and large activity network, the manufacturing scheduling problems are frequently based on simple and few precedence relations between the tasks that are to be planned (cf. Lee, 2002). Therefore, the majority of these methods are merely marginally suitable for solving large-scale and complex project-scheduling problems. For solving decentralised multi-project scheduling problems, only a few methods are available (cf. Lee, 2002; Confessore et al.,

2007; Homberger, 2007; Wauters et al., 2010). Each of these four groups of researchers proposed a MAS solution, in which multiple agents — corresponding to the decision makers (such as project managers) in the real multi-project environment — collaborate, guided by some coordination mechanism, to construct and select viable schedules².

Below, we discuss briefly the four existing MAS solutions, and compare the solution methods in Table 3.2.

Authors	Agent Model			Intra-project scheduling method	Coordination mechanism
	project (activity) agents	resource agents	mediator agents		
Lee (2002)	fine-grained	yes	yes		combinatorial auction
Confessore et al. (2007)	coarse-grained	no	yes	p-SGS & LFT	iterative combinatorial auction
Homberger (2007)	coarse-grained	no	yes	RES	negotiation
Wauters et al. (2010)	coarse-grained	no	yes	learning automata	dispersion game & s-SGS

Table 3.2: MAS-based solution methods for DRCMPSP

Lee (2002)

Lee (2002) developed a MAS for decentrally planning a dynamic variant of the DRCMPSP (see also Lee et al., 2003). The model is based on the assumption that several companies or decision makers have to distribute their locally available resources to projects, which appear dynamically during the planning horizon (i.e., equivalent to the variable-project-release-time problem, see §2.2.3). The core of the MAS scheduling framework is formed by three types of agents: (1) resource agents, (2) activity agents, and (3) a coordinator agent. A resource agent manages a single resource of a type in one department. An activity agent is responsible for scheduling an individual activity. The coordinator agent creates a marketplace where the resource-agent-managed resource capacities can be offered for sale to the activity agents. The sale is handled through *combinatorial auctions*. We note that the approach of modelling a resource agent for a single resource and modelling an activity agent for an individual activity is ‘fine-grained’.

²A recent work by Araújo et al. (2009) also claims to have found a MAS solution for multi-project scheduling problems. However, the problem they study is not quite a project-scheduling problem. Below, we list three properties of their problem which clearly show the difference with our problem. First, an activity in their problem only requires a single unit of resource. Second, the activities of a project in their problem is arranged in a sequence, rather than in a network. Third, each resource type defined in their problem only has a single unit. These three properties render their problem a machine-scheduling problem. In addition, their objective is on project portfolio level rather than on operational level.

Confessore et al. (2007)

Confessore et al. (2007) modelled two types of agents to deal with DRCMPSPs. The two agent types are (1) project agents and (2) a coordinator agent. A project agent is responsible for scheduling one project comprising more than one activity. There are as many project agents as projects. This modelling approach is ‘coarse-grained’ compared to the ‘fine-grained’ model proposed by Lee (2002). For scheduling a project, the corresponding project agent uses a plain construction heuristic that consists of a *parallel schedule generation scheme* (p-SGS) and a *minimum latest finish time first* (LFT) priority rule. The coordinator agent is responsible for allocating the shared resources to the project agents. The resource allocation is carried out via a combinatorial auction. Instead of using a common objective, each project agent aims at minimising its makespan while the coordinator agent declares the auction winners when the project agents compete for the shared resources. The evaluations of MAS solution quality are carried out on several multi-project instances, with up to 5 projects and 18 activities.

Homberger (2007)

In conformity with Confessore et al. (2007), Homberger (2007) adopted a ‘coarse-grained’ decomposition of the multi-project problem over agents (i.e., there are as many “scheduler agents” as projects). Each scheduler agent is responsible for making schedules for one project and a “mediator agent” is created to coordinate the resource allocations amongst the scheduler agents. The scheduling procedure is carried out as follows. (1) The mediator agent initially allocates resource capacities to each of the projects in a way that every single project is feasibly solvable. (2) Each scheduler agent adopts a *restart evolution strategy* (RES) (i.e., an evolutionary strategy that adaptively ‘learns’ a better project schedule, and a ‘restart’ procedure that helps to escape from local optima) to construct and learn a (near) optimal schedule based on the allocated resource capacities. (3) The resource capacities allocated but unused by a project are sent back to the mediator agent. (4) The mediator agent collects and aggregates all the unused resource capacities and makes them observable to all scheduler agents. (5) Scheduler agents calculate possible schedule improvement by RES, based on the current schedule plus the unused resource capacities. (6) Improvements are sent to the mediator agent, who then selects the biggest improvement and updates the pool of unused capacities. Steps 5 and 6 are repeated until no improvement can be found. The solution methods that Homberger (2007) proposed are capable of solving problems with up to 20 projects and 120 activities each, and the performances are claimed to be competitive with a central solution using the RES.

Wauters et al. (2010)

Similar to the model proposed by Homberger (2007), the MAS model introduced by Wauters et al. (2010) consists of two classes of agents — project agents and one mediator agent. Instead of making schedules for individual activities, the project agents construct an *activity list*. They do so by employing a network of straightforward reinforcement learning devices called *learning automata*. Coordination is achieved by a sequence game (i.e., a dispersion game) in which each project agent submits its activity list and learns a

suitable place in the overall sequence of activity lists of all projects. The sequence game is played using a probabilistic version of the *basic simple strategy* (BSS). The overall schedule of all project activities is built by the mediator agent using a *serial schedule generation scheme* (s-SGS). Wauters et al. (2010) use GT-MAS to denote their solution method. GT-MAS stands for a game-theoretic MAS scheduling approach. GT-MAS produces so far the best results in terms of average project delay (APD) in the Multi-project Scheduling Problem Library (MPSPLib).

3.2.3 Towards a New MAS Solution

Multiagent systems offer many desirable properties (cf. Durfee and Rosenschein, 1994; Weiss, 2000), such as (1) *self-interestedness*, (2) *flexibility*, (3) *scalability*, (4) *efficiency*, (5) *robustness*, (6) *reliability*, (7) *cost effectiveness*, and (8) *reusability*. Within these desirable properties, properties 1 to 3 are related to agent model and mechanism design, properties 4 and 5 involve the scheduling approaches employed by the agents, and properties 6 to 8 apply to the software development of the system³.

In this subsection, we review the agent models and mechanisms in the aforementioned four existing MAS solution methods and discuss their shortcomings with respect to the three properties — (a) *self-interestedness*, (b) *flexibility* and (c) *scalability*. The discussion will lead to a new agent-based model for the AGH scheduling problem. The desired properties 4 and 5 will be further investigated in Chapter 5 and 6.

A: Self-interestedness

Self-interestedness indicates that an agent concerns for its own advantage and well-being in a multiagent system.

Three out of the four MAS models (i.e., the models proposed by Confessore et al. (2007), Homberger (2007), and Wauters et al. (2010)) ignore totally the self-interestedness of resource managers. In many modern industrial project-scheduling environments, resources shared by the projects are managed by third-party resource providers (or resource managers). The resource managers often have objectives different from those of the project managers with respect to the resource allocations. In addition, within all resource managers in a scheduling problem, one may have different objective(s) from the others. Therefore, a more realistic MAS model has to take into account the self-interested nature of resource managers in a project-scheduling environment.

B: Flexibility

Flexibility means the ability of incorporating new agents into the system without affecting the operability of the other agents.

Agent coordinations in all of the four MAS solution methods (auction ($\times 2$), negotiation, dispersion game) are achieved in large-scale **synchronous** manners. The three coordination mechanisms can be characterised as follows. (1) In the auction mechanisms proposed by Lee (2002) and Confessore et al. (2007), an auction calls for a collection

³In this thesis, we focus on the desired properties 1 to 5 of a MAS system. Software development is beyond the scope of this thesis.

of bids from a large number of agents. (2) In the negotiation mechanism proposed by Homberger (2007), a schedule improvement requires a collection of improvement proposals from a large number of agents. (3) In the dispersion game for coordination proposed by Wauters et al. (2010), the participation of all project agents takes place at the same time. Large-scale synchronisation restricts new agents from easily joining in the system. In order to increase flexibility, **asynchronous** coordination should be explicitly taken into account in agent-based models (Boer et al., 2007).

C: Scalability

Scalability refers to the ability of accommodating an increased number of agents in the system.

Thanks to the coarse-grained modelling approach, three out of the four MAS solutions (i.e., the solutions by Confessore et al. (2007), Homberger (2007), and Wauters et al. (2010)) are able to scale to large problem instances. Although Lee (2002) models resource managers as self-interested agents, the proposed fine-grained model, where each activity is modelled as an agent, creates easily a large number of agents when the multi-project problem concerns a large number of activities. This increases significantly inter-agent communication loads and synchronisation efforts especially when using auction as coordination mechanism. Methods in a fine-grained model can only solve small problem instances (with up to 9 projects and 15 activities in their experiments). Therefore, a more scalable MAS model has to employ a coarse-grained model in order to scale up with a large problem instance.

To summarise, designing an effective agent-based model for DRCMPSP requires for (1) a coarse-grained modelling approach to encapsulate the agents, (2) a model that reflects the self-interestedness of all agents, (3) an asynchronous coordination mechanism that involves only a small number of agents at a time. In Chapter 4, we design an novel agent-based model for decentralised multi-project scheduling problem. The model includes two classes of self-interested agents, as well as an asynchronous lease-based coordination mechanism.

3.3 Project Scheduling under Uncertainty

In this section we survey approaches for scheduling projects under uncertainty. In general, we distinguish five classes of such approaches: (1) *proactive-reactive scheduling*, (2) *stochastic scheduling*, (3) *fuzzy scheduling*, (4) *contingent scheduling*, and (5) *sensitivity analysis*. The survey of each class is by no means exhaustive. It only intends to give the reader a feeling of the research directions up to date.

3.3.1 Proactive-reactive Scheduling

Proactive-reactive scheduling is a two-stage scheduling approach. First, prior to the project start, *proactive scheduling* constructs a *baseline schedule* (a.k.a. *predictive schedule* or *pre-schedule*) by employing an exact or a heuristic method. Then, while executing

the baseline schedule, *reactive scheduling* revises or reoptimises the baseline schedule at hand when an unexpected event occurs.

In case a proactive baseline schedule is developed using a deterministic scheduling method without any anticipation of variability, the obtained optimal or sub-optimal baseline schedule will account for intensive reactive scheduling efforts. Every time a tiny schedule disruption occurs, the proactive baseline schedule might be invalidated. To reduce the effort of reactive scheduling, *robust proactive scheduling* aims at constructing a baseline schedule that incorporates anticipated project-execution variability prior to the project start. Generating a robust proactive schedule prior to the project start may significantly reduce the reactive-scheduling efforts. However, it should be observed that a proactive scheduling will always require a reactive component to deal with schedule disruptions that cannot be absorbed by the baseline schedule.

An exhaustive investigation and empirical study of proactive-reactive project scheduling can be found in van de Vonder et al. (2007).

3.3.2 Stochastic Scheduling

Stochastic scheduling is concerned with modelling operational uncertainty by a probabilistic random distribution of activity processing times. The stochastic scheduling methodology basically views the project scheduling problem as a multi-stage decision process. Scheduling policies are used to define which activities are to be started at random decision points over time, based on the observed and the a priori knowledge about the processing time distribution.

The literature on stochastic project scheduling with resource constraints is rather sparse. The larger body of theoretical work has been produced by a limited number of researchers (cf. Demeulemeester and Herroelen, 2002; Bonfill-Teixidor, 2006). By ‘stochastic project scheduling’ it is understood that (renewable) resource constraints are imposed; this is to be contrasted with the Program Evaluation and Review Technique problem (PERT-problem), which is resource-unconstrained and to which a large amount of literature has been devoted.

3.3.3 Fuzzy Scheduling

An alternative of modelling operational uncertainty into a probabilistic random distribution of activity processing times is to use a possibility-based model (fuzzy model). There are several ways to fuzzify scheduling problems. Two important approaches are (1) fuzzy activity processing time, and (2) fuzzy due time (Słowiński and Hapke, 1999). The researchers advocating the fuzzy activity processing time approach argue that probability distributions for the activity processing times are unknown due to the lack of historical data. As activity processing times have to be estimated by human experts, in a non-repetitive or even unique setting, project management is often confronted with judgements that are rather vague and imprecise. For example, the processing time of an activity is clearly more than two days and less than five days; about three days is usual. In those situations, which involve imprecision rather than uncertainty, the fuzzy set scheduling literature recommends the use of fuzzy numbers for modelling activity processing times, rather than stochastic variables. Instead of probability distributions, these

quantities make use of membership functions, based on *possibility theory* (Herroelen and Leus, 2005).

3.3.4 Contingent Scheduling

The contingent scheduling approach is based on the generation of multiple baseline schedules or baseline schedule fragments before and/or during project execution that either optimally respond to anticipated disruptive events or are equivalent in performance (cf. Herroelen and Leus, 2005). If a disruption effectively takes place, an adequate reaction is simply to switch to the applicable schedule (fragment), or to shift to an equivalent schedule compatible with the disruption. This approach focuses on flexibility rather than robustness and is especially valuable for time-critical reactive scheduling (cf. Herroelen and Leus, 2005).

3.3.5 Sensitivity Analysis

A number of recent research efforts focus on the sensitivity analysis of machine-scheduling problems (Mauroy et al., 1997; Hall and Posner, 2004). Sensitivity analysis is the study of how the uncertainty in the output of a mathematical model can be apportioned, qualitatively or quantitatively, to different sources of variation in the input of the model. Sensitivity analysis addresses “What if ...?” types of questions that arise from parameter changes. The authors Hall and Posner (2004) study polynomially solvable and intractable machine-scheduling problems and try to provide answers to a number of fundamental questions. Below we list six of them.

1. *What are the limits to the change of a parameter such that the solution remains optimal?*
2. *Given a specific change of a parameter, what is the new optimal cost?*
3. *Given a specific change of a parameter, what is a new optimal solution?*
4. *When does the objective function value remain optimal?*
5. *What types of sensitivity analysis are useful to evaluate the robustness of optimal solutions?*
6. *What types of sensitivity analysis can be performed without using the full details of the solution?*

The use of sensitivity analysis for general decision-making under uncertainty has been the subject of critique. We refer the reader to Wallace (2000), who stresses that flexibility options are not appropriately recognised when using deterministic models with sensitivity analysis. Wallace (2000) points out that the technique is appropriate only when analysing an allowable variation in controllable parameters.

3.4 Chapter Summary

Numerous techniques have been employed to address the problems of project scheduling. In this chapter, we reviewed the existing solution methods with emphases on the problems

in which (1) multiple projects are presented, (2) resources and projects are managed by different autonomous decision makers, and (3) project executions are carried out in an environment with various uncertainty. For clarity, we summarise them below together with our conclusions.

First, we discussed centralised solution methods that deal with RCMPSP in a deterministic setting (see Section 3.1). The investigations have shown that (1) RCMPSPs are strongly NP-hard problems and exact methods are impractical to be deployed for solving real-world problems, and (2) heuristics and meta-heuristics are widely used in practice.

Second, we focused on investigating decentralised solution methods for problems with decentralised decision makers (see Section 3.2). We discussed in detail four recent investigations (see Lee, 2002; Confessore et al., 2007; Homberger, 2007; Wauters et al., 2010) and outlined the drawbacks of each of them in dealing with problems with autonomous resource managers. The main issue here is that the managers are self interested.

Third, we discussed existing solution models that deal with uncertainty in scheduling problems (see Section 3.3). We listed five classes of approaches and discussed their applicabilities in AGH scheduling problems.

Chapter 4

A Lease-based Multiagent Model

AGH scheduling problems fall into a category of scheduling problems with information asymmetry and decentralised decision-making processes (see Section 2.2). Solving such a category of scheduling problems requires models and techniques that take into account the strategic behaviour of individual decision makers (cf. Heydenreich et al., 2006). In DAI, multiagent systems are known for being capable of dealing with problems with inherent informational and managerial decentralisation (cf. Bonabeau, 2002; Shoham and Leyton-Brown, 2009). Hence, it is appropriate to model an AGH scheduling problem in an agent-based model and to solve it by a multiagent system. This chapter addresses our first research question, which reads as follows.

RQ1: *How can an AGH scheduling problem be represented in an agent-based model?*

In Section 3.2, we listed several earlier attempts of employing agent-based systems for modelling and solving DRCMPSPs (Lee et al. (2003), Confessore et al. (2007), Homberger (2007), and Wauters et al. (2010)). We have shown that the agent-based models in their systems cannot adequately represent an AGH scheduling problem. The inadequateness stems from potential communication overload caused by the ‘fine-grained’ model to non-self-interested resource-type managers. This means that the associated multiagent scheduling systems are inappropriate for solving AGH scheduling problems (see §3.2.3). In this chapter, we propose a novel agent-based model that serves as the solution framework for dealing with AGH scheduling problems.

The proposed agent-based model comprises two modelling components: (1) *modelling agent representations*, which involves choosing a proper agent-encapsulation approach to model agents, and (2) *modelling agent interactions*, which involves designing a mechanism that coordinates the agent decisions. In the first two sections of the chapter we address the two components, respectively.

In Section 4.1, we propose a role-based agent model that adopts a physical-entity-oriented encapsulation approach. Herein, we distinguish between two classes of agents,

i.e., resource agents and project agents. Each agent out of one of the two classes represents a physical entity or an organisation in the real world.

In Section 4.2, we introduce a *lease-based market mechanism* for coordinating agent decisions. In the mechanism, the scheduling decisions over a single activity are coordinated in a *lease-based slot negotiation scenario*. The scenario involves decisions of a project agent and several resource agents.

Section 4.3 concludes the chapter by answering RQ1.

4.1 Agents, Schedules, and Utilities

The first main issue in agent-based modelling is to define accurately and properly the individual agents' roles in a system. This issue depends on the decision what an agent should encapsulate, namely, the decision on what precisely represents an agent (Lee et al., 2003). Many encapsulation approaches exist for problems in a variety of situations. Among these approaches, most of them fall into three major categories (cf. Shen and Norrie, 1999): (1) the category of *function-oriented* approaches, (2) the category of *physical-entity-oriented* approaches, and (3) the category of *hybrid* approaches,

In function-oriented approaches, agents are used to encapsulate some functions such as task decomposition, activity coordination, conflict detection, and conflict resolution. Agents defined in this category are referred to as *task* agents.

In physical-entity-oriented approaches, agents represent physical entities or organisations such as managers, workers, machines, and components; agents defined in the second category are referred to as *representative agents*.

In hybrid approaches, both task agents and representative agents are employed. All the four existing MAS solutions to DRCMPSP use hybrid modelling approaches. Examples of task agents are the coordinator agents in Lee (2002) and Confessore et al. (2007), the mediator agents in Homberger (2007) and Wauters et al. (2010). Examples of representative agents are the activity agents and the resource agents in Lee (2002), the project agents in Confessore et al. (2007), Homberger (2007), and Wauters et al. (2010).

The physical-entity-oriented approaches feature the self-interested nature of agents. Plus, the approaches naturally define distinct sets of state variables that can be managed efficiently by individual agents. Therefore, the approaches of the second category are appropriate for modelling the project-scheduling environment, where more physical entities are involved compared to transaction-oriented information-system domains (cf. Lee et al., 2003).

In DRCMPSPs, different resource types are managed by different resource-type managers. Similarly, different projects are managed by different project managers. These (resource-type and project) managers are self-interested parties. They make scheduling decisions autonomously with respect to their own objectives. Taking into account the AGH-specific project-scheduling environment, we adopt the category of physical-entity-oriented approaches. We will model resource-type managers and project managers as two classes of agents, namely *resource agents* and *project agents* (see Figure 4.1).

In the following two subsections (i.e., §4.1.1 and §4.1.2) we specify the roles, schedules, and utilities of the two agent classes.

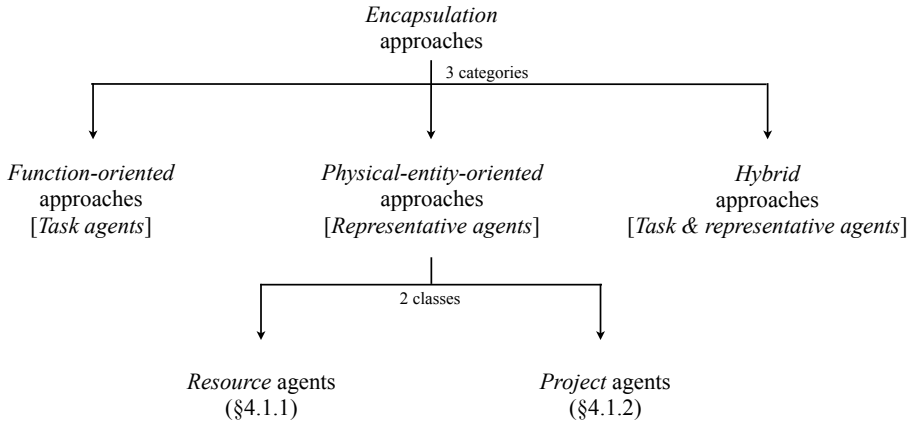


Figure 4.1: Agent encapsulation approaches

4.1.1 Resource Agent, Schedule, and Utility

We model a first class of agents, namely **resource agents**, as the representatives of the resource-type managers in a DRCMPSP. In the following, we define the role, the schedule, and the utility of a resource agent.

Resource Agent

Let RA_k denote the resource agent that is responsible for making schedules of resource type R_k . The system includes as many resource agents as resource types, so $|RA_k| = |R_k| = K$ (see §2.2.4).

In an AGH-specific DRCMPSP, a ground-service provider (e.g., a fuelling company) can be seen as a resource-type manager. Hence, in our model, a resource agent represents a ground-service provider. The role of a resource agent is managing the set of resources of the same type (e.g., fuelling trucks) and makes the corresponding scheduling decisions in order to maximise its utility.

Knowing the role of a resource agent, below we discuss what constitutes a resource-agent schedule.

Resource-agent Schedule

A schedule of a resource agent specifies for each of the activities, which request the use of the resources managed by the resource agent, **when** and **how many** resources are allocated to the activity.

An allocation can be represented by a tuple of three elements: (1) an activity, (2) a time interval, and (3) a resource capacity of the resource type managed by the resource agent. Below, we define the tuple as a *resource-agent slot* (see Definition 4.1).

DEFINITION 4.1 Resource-agent Slot. A resource-agent slot is a tuple $\pi_{i,j,k}^R = \langle A, I, R \rangle$, in which

- $A = a_{i,j}$ is an activity;
- $I = [t_s, t_e)$ is a time interval specified by a start time t_s (inclusive) and an end time t_e (exclusive);
- $R = (k: c)$ is the allocated resource capacities, defined by the resource type R_k and a resource capacity c .

Figure 4.2 depicts a resource-agent slot $\pi_{i,j,k}^R$ on the timeline of the resource agent RA_k .

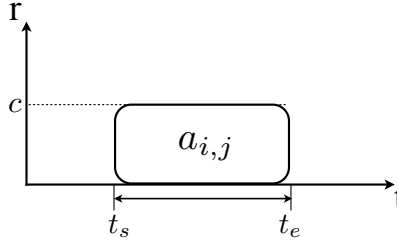


Figure 4.2: A resource-agent slot $\pi_{i,j,k}^R$

Resources of one type may be used by different activities from different projects. We define a *resource-agent schedule* as a set of resource-agent slots, each of which involves one activity (see Definition 4.2).

DEFINITION 4.2 Resource-agent Schedule. A resource-agent schedule Π^k of resource agent RA_k is a set of resource-agent slots: $\Pi^k = \{\pi_{i,j,k}^R \mid 1 \leq i \leq m \wedge 1 \leq j \leq n_i \wedge r_{i,j}^k \in \mathbb{N}^+\}$.

Figure 4.3 shows an example of a resource-agent schedule Π^k consisting of six slots ($\Pi^k = \{\pi_1, \pi_2, \dots, \pi_6\}$). The dotted line in this figure indicates that the resource type R_k is constrained by a maximum capacity \bar{c}_k (see resource-capacity constraint in §2.1.3). Given a resource-agent schedule Π^k , we can derive for each point in time the amount of resources of R_k that are used by the scheduled activities, the so-called *resource load*. We define resource load as follows.

DEFINITION 4.3 Resource Load. Given a schedule Π^k of a resource agent RA_k , the resource load λ of R_k at time point t is a function $\lambda: \Pi \times T \rightarrow \mathbb{N}$, where

$$\lambda(\Pi^k, t) = \sum_i \sum_j c(\pi_{i,j,k}^R), \quad \pi_{i,j,k}^R \in \Pi^k \wedge t \in \pi_{i,j,k}^R$$

in which $c(\pi)$ is the allocated resource capacity of slot π (see Definition 4.1), and $t \in \pi_{i,j,k}^R$ stands for $t \in I \wedge I \in \pi_{i,j,k}^R$.

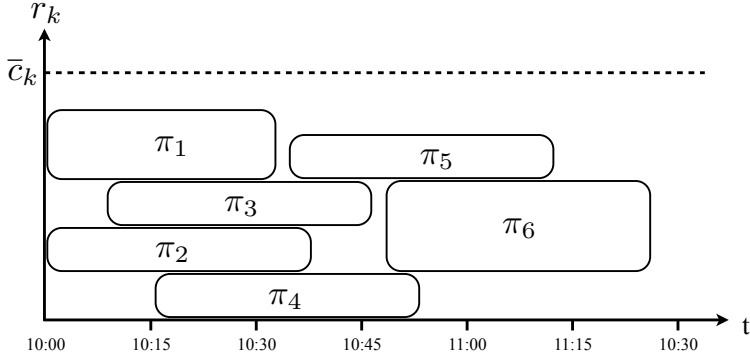


Figure 4.3: An example of a resource-agent schedule with six slots

The resource load should not exceed the maximum capacity of a resource type at any time t , as the resource-capacity constraint should always be satisfied. We define a *feasible* resource-agent schedule with resource-capacity constraint as follows.

DEFINITION 4.4 A Feasible Resource-type Schedule. A schedule Π^k of a resource-agent RA_k is called *feasible* when Π^k satisfies the following resource-capacity constraint:

$$\lambda(\Pi^k, t) \leq \bar{c}_k, \quad t \geq 0.$$

The primary task of a resource agent during scheduling is to construct a feasible resource-agent schedule. Furthermore, if a resource agent has a choice among many feasible resource-agent schedules, it would prefer a certain schedule to other schedules. The preference can be measured by a utility function. In the following, we model the utility of a resource agent.

Resource-agent Utility

Modelling resource-agent utilities addresses the self-interested nature of the resource agents. In a DRCMPSP, a self-interested resource agent representing a resource-type manager is interested in allocating its own resources in the best way. Accordingly, resource-based objectives are often considered by the resource agents while making scheduling decisions.

A resource agent may consider a specific resource-based objective or a combination of various resource-based objectives. Examples of resource-based objectives for a single resource agent are *minimising the resource procurement cost* (i.e., a resource-investment objective) and *minimising the resource utilisation variation* (i.e., a resource-levelling objective). Here, we choose a resource-levelling objective as an example objective to study the utility of a resource agent. We note that choosing the resource-levelling objective is arbitrary. The resource agents in the eventual MAS scheduling system are not restricted to using such a particular objective only.

A resource-levelling objective strives for minimising the resource utilisation variation over time. This objective is often achieved by *minimising the sum of squared resource utilisation costs*, formulated as follows.

$$\begin{aligned} \text{Find} \quad & \Pi^k = \arg \min_{\Pi^k} f(\Pi^k), \\ \text{where} \quad & f(\Pi^k) = c_k^u \cdot \sum_{t \geq 0} \lambda^2(\Pi^k, t). \end{aligned} \quad (4.1)$$

We recall that c_k^u stands for the utilisation cost per unit of resource type R_k per unit of time (see §2.1.4). It is important to notice that the objective function in Equation 4.1 is different from the one in Equation 2.15, where the latter minimises the overall costs of all resource types. Instead, the objective here is only concerned with minimising the cost of a single resource type (i.e., R_k).

Since the objective of resource agent RA_k is to **minimise** $f(\Pi^k)$, we can model a *utility* (performance measure of a given schedule) of RA_k being the negative value of $f(\Pi^k)$. So, we have for resource agent RA_k , a resource-levelling utility function:

$$U_{RA_k}(\Pi^k) = -c_k^u \cdot \sum_{t \geq 0} \lambda^2(\Pi^k, t). \quad (4.2)$$

4.1.2 Project Agent, Schedule, and Utility

In our proposed agent-based model for AGH scheduling problems, the second class of agents contains the **project agents** (see Figure 4.1). This subsection contains the model of project agent, project-agent schedule and project-agent utility.

Project Agent

A project agent represents a project manager that is responsible for making scheduling decisions for the corresponding project. Let PA_i denote the project agent representing the project manager of P_i . There are as many project agents as projects in the system, so $|PA_i| = |P_i| = m$ (see §2.2.4).

In an AGH-specific DRCMPSP, a project refers to an aircraft turnaround process, a project agent can be seen as a turnaround manager. It manages a set of ground-handling operations and makes the corresponding scheduling decisions to achieve certain turnaround objectives.

Using a project agent to make scheduling decisions for all activities of the project is a ‘coarse-grained’ agent-based modelling approach (see §3.2.2). The ‘coarse-grained’ approach is in conformity with Confessore et al. (2007), Homberger (2007) and Wauters et al. (2010). We have chosen to use the ‘coarse-grained’ approach for the following two reasons: (1) modelling each project manager by a software agent matches the self-interested nature of agents; (2) a ‘fine-grained’ modelling approach in which each activity is represented by a software agent (see Lee, 2002) may create a vast amount of agents, which as a consequence may cause an overload of inter-agent communications when a project comprises a large number of activities.

Project-agent Schedule

In the scheduling phase, a project agent is responsible for finding for each of its activities a schedule containing the required resource types. In case an activity requires resources from more than one resource type, a schedule of an activity should comprise more than one reservation. Each of the reservations is concerned with one resource type. Similar to resource-agent slot, a reservation can be represented by a tuple of three elements: (1) a time interval, (2) a resource requirement from a resource type, and (3) an activity. Below, we define the reservation tuple as a project-agent slot (see Definition 4.5).

DEFINITION 4.5 Project-agent Slot. *A project-agent slot is a tuple $\pi_{i,j,k}^P = \langle A, I, R \rangle$, in which*

- $A = a_{i,j}$ is an activity;
- $I = [s_{i,j}, s_{i,j} + p_{i,j})$ is a time interval specified by start time $s_{i,j}$ (inclusive) and duration $p_{i,j}$;
- $R = (k: r_{i,j}^k)$ is a resource requirement defined by resource type R_k and required amount $r_{i,j}^k$.

Subsequently, we define an *activity schedule* (see Definition 4.6).

DEFINITION 4.6 Activity Schedule. *A schedule of an activity $a_{i,j}$ is a set of project-agent slots $\Pi_{i,j} = \{\pi_{i,j,k}^P \mid k \in \{1, \dots, K\} \wedge r_{i,j}^k \in \mathbb{N}^+\}$.*

Figure 4.4 shows an activity schedule $\Pi_{i,j}$ on the timeline of a project agent PA_i . In the figure, we notice that all slots in $\Pi_{i,j}$ have an identical time interval. This is because all slots in $\Pi_{i,j}$ are related to the same activity (i.e., $a_{i,j}$), the time intervals of the slots should be identical¹. Let $\pi \stackrel{I}{=} \pi'$ denote the fact that two slots have the same time interval $I = I'$, where $I \in \pi$ and $I' \in \pi'$. For any slots in an activity schedule, the following equation holds.

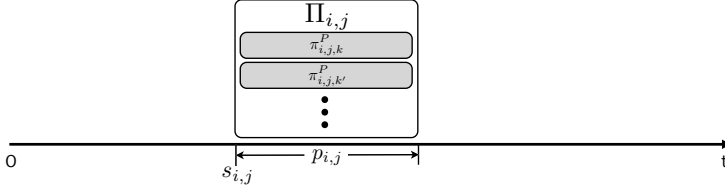
$$\pi_{i,j,k}^P \stackrel{I}{=} \pi_{i,j,k'}^P, \quad \forall \pi_{i,j,k}^P, \pi_{i,j,k'}^P \in \Pi_{i,j}. \quad (4.3)$$

Having defined an activity schedule in our agent-based model for DRCMPSP, we define a *project-agent schedule* as follows.

DEFINITION 4.7 Project-agent Schedule. *A schedule of a project agent PA_i is a set of activity schedules $\Pi_i = \{\Pi_{i,1}, \Pi_{i,2}, \dots, \Pi_{i,n_i}\}$, where $\Pi_{i,j}$ is the schedule of activity $a_{i,j}$ ($a_{i,j} \in A_i$).*

We recall that activities of a project are bound by precedence constraints and the project may also have additional temporal constraints, such as a project-release-time constraint and a project-deadline constraint (see §2.1.2). Below, we define a *feasible project-agent schedule* (see Definition 4.8).

¹Slot-to-slot relations are the same as interval-to-interval relations (see Figure 2.2). Likewise, the possible temporal relations between a time point t and a slot π are the same as the point-to-interval relations (see Figure 2.3).

Figure 4.4: An activity schedule $\Pi_{i,j}$

DEFINITION 4.8 A Feasible Project-agent Schedule. A schedule Π_i of a project agent PA_i is called feasible when Π_i satisfies all precedence constraints as well as all additional temporal constraints:

$$\begin{aligned} \pi_{i,j,k}^P &\stackrel{I}{\leq} \pi_{i,l,k'}^P, & a_{i,j} &\prec a_{i,l}, \forall \pi_{i,j,k}^P, \pi_{i,l,k'}^P \in \Pi_i, \\ \overline{rl}_i &\stackrel{I}{\leq} \pi_{i,j,k}^P \stackrel{I}{\leq} \overline{dl}_i, & \forall \pi_{i,j,k}^P &\in \Pi_i, \end{aligned} \quad (4.4)$$

where $\pi \stackrel{I}{\leq} \pi'$ indicates $t_e(\pi) \leq t_s(\pi')$.

Similar to resource agents, a project agent having a choice among many feasible project-agent schedules would prefer one feasible schedule to other feasible schedules. In the following, we model the utility of a project agent.

Project-agent Utility

As discussed in §2.2.4, in the AGH-scheduling environment, a predefined time window at a terminal gate (or at a remote stand) is assigned to an aircraft. The ending time of the gate assignment is also the scheduled departure time of the aircraft for the next flight. Thus, the departure time of an aircraft can be seen as the *due time* of its turnaround process. A delayed departure is often punished by a delay penalty. As a consequence, aircraft managers are trying to schedule all their ground-handling operations within the gate-assignment time windows.

As mentioned in the resource-agent utility modelling, the resource utilisation costs may vary from time to time. As a direct consequence, the price of receiving a ground service varies as well. In this case, aircraft managers are trying to find the services with relatively cheap prices for carrying out their ground-handling operations, while keeping an eye on avoiding any departure delay. Hence, we have chosen for each project agent a combination of two objectives: (1) minimising the project delay cost (i.e., $c_i^{dl} \cdot dl_i(\Pi_i)$) and (2) minimising the total service costs of its all ground-handling operations (i.e., $\sum_{\pi_{i,j,k}^P \in \Pi_i} rc(\pi_{i,j,k}^P)$).

$$\begin{aligned} \text{Find } \Pi_i &= \arg \min_{\Pi_i} f(\Pi_i), \\ \text{where } f(\Pi_i) &= c_i^{dl} \cdot dl_i(\Pi_i) + \sum_{\pi_{i,j,k}^P \in \Pi_i} rc(\pi_{i,j,k}^P). \end{aligned} \quad (4.5)$$

Like in the resource-agent utility modelling, the utility function of a project agent PA_i is the negative value of $f(\Pi_i)$:

$$U_{PA_i}(\Pi_i) = -c_i^{dl} \cdot dl_i(\Pi_i) - \sum_{\pi_{i,j,k}^P \in \Pi_i} rc(\pi_{i,j,k}^P) \quad (4.6)$$

In the equations above, $dl_i(\Pi_i)$ denotes the project delay given a schedule Π_i (see Equation 2.12). c_i^{dl} denotes the delay cost per time unit for P_i . The function $rc(\pi_{i,j,k}^P)$ stands for the resource cost of the project-agent slot $\pi_{i,j,k}^P$.

The resource cost of a project-agent slot is set by a resource agent. In order to create an incentive for project agents to reserve the slots that do not create resource utilisation peaks, the resource agents will use a utility-decomposition technique to determine the price of a slot. The detailed cost analyses will be discussed in Section 4.2.

4.1.3 A Conflict-free and Feasible Agent-based AGH Schedule

Earlier in this section, we defined a resource-agent schedule as well as a project-agent schedule (see Definition 4.2 and 4.7). Below, we define an agent-based AGH Schedule.

DEFINITION 4.9 An Agent-based AGH Schedule. *An agent-based AGH schedule S is a complete set of all agent schedules.*

$$S = \{S^R, S^P\} \quad (4.7)$$

In Equation 4.7, S consists of two sets: (1) a complete set S^R of all resource-agent schedules ($S^R = \{\Pi^k | k \in \{1, \dots, K\}\}$) and (2) a complete set S^P of all project-agent schedules ($S^P = \{\Pi_i | i \in \{1, \dots, m\}\}$).

In order to retain an efficient solution to AGH scheduling problems an agent-based AGH schedule need to be at least *conflict free*. Below we define a conflict-free agent-based AGH schedule.

DEFINITION 4.10 An Conflict-free Agent-based AGH Schedule. *An agent-based AGH schedule S is conflict free, when the following holds.*

$$S^R = S^P, \quad S^R, S^P \in S \quad (4.8)$$

When an activity $a_{i,j}$, of which the processing mode $\mu_{i,j} = \langle \{(k: r_{i,j}^k) | k \in \{1, \dots, K\} \wedge r_{i,j}^k \in \mathbb{N}^+\}, p_{i,j} \rangle$, on a resource type R_k is scheduled, two slots are created: (1) a resource-agent slot $\pi_{i,j,k}^R$ and (2) a project-agent slot $\pi_{i,j,k}^P$. Subsequently, the resource-agent slot $\pi_{i,j,k}^R$ is included in S^R ($\pi_{i,j,k}^R \in S^R$), and the project-agent slot $\pi_{i,j,k}^P$ is included in S^P ($\pi_{i,j,k}^P \in S^P$). Since the two slots concern a same task (i.e., processing activity $a_{i,j}$ with $r_{i,j}^k$ amount of resource type R_k in a time interval I), a conflict-free schedule proclaims the equivalence of the two slots: $\pi_{i,j,k}^R = \pi_{i,j,k}^P$. The equivalence implies that for any of the slots in S^R there is an equivalent slot in S^P , and vice versa.

Furthermore, an efficient solution to AGH scheduling problems requires an agent-based AGH schedule to be at least *feasible*. Combining the definition of a feasible resource-agent

schedule and the one of a feasible project-agent schedule, we can define a feasible agent-based AGH schedule as follows.

DEFINITION 4.11 A Feasible Agent-based AGH Schedule. *An agent-based AGH schedule S is called feasible when none of the resource-capacity constraints, precedence constraints, and additional temporal constraints are violated.*

$$\begin{aligned}
 \lambda(\Pi^k, t) &\leq \bar{c}_k, & t &\geq 0, \forall \Pi^k \in S^R \\
 \pi_{i,j,k}^P &\stackrel{I}{\leq} \pi_{i,l,k'}^P, & a_{i,j} &\prec a_{i,l}, \forall \pi_{i,j,k}^P, \pi_{i,l,k'}^P \in \Pi_i, \forall \Pi_i \in S^P \\
 \bar{r}l_i &\stackrel{I}{\leq} \pi_{i,j,k}^P \stackrel{I}{\leq} \bar{d}l_i, & \forall \pi_{i,j,k}^P &\in \Pi_i, \forall \Pi_i \in S^P
 \end{aligned} \tag{4.9}$$

In Section 4.2, we discuss how scheduling decisions of resource agents and project agents can be coordinated in order to achieve such a conflict-free and feasible agent-based AGH schedule.

4.2 Lease-based Market Mechanism

Scheduling the equivalence of a pair of slots ($\pi_{i,j,k}^R$ and $\pi_{i,j,k}^P$) involves decisions of two agents of different classes — a resource agent RA_k and a project agent PA_i . Inevitably, the decisions about the slots, such as the start times and finish times of the slots' time intervals, cannot be decided unilaterally by any one of the agents. The resource agent will make sure that the slot is not violating the resource-capacity constraint and the project agent will keep an eye on avoiding violation of precedence constraints and additional temporal constraints. Moreover, when an activity requires more than one resource type, the project agent also has to make sure that all project-agent slots of the activity schedule have the same time interval (see Equation 4.3). Since the agents are autonomous decision makers, the scheduling decisions of the agents have to be coordinated.

In this section, we describe a *lease-based market mechanism*. In the mechanism, resources of a certain type are regarded as merchandise or products owned by the resource agent. A project agent **leases** a certain amount of resources for a period of time from a resource agent to process one of its activities. A lease is a mutual commitment between a project agent and resource agent. Since a lease (1) involves processing an activity, (2) requires an amount of resources, and (3) lasts a period of time, it can be represented in the form of a slot.

Since both the resource agent and the project agent are self-interested and autonomous decision makers, they must have their own value systems on pricing any single slot. However, in the utility modelling of both classes of agents, the utilities are only concerned with the entire agent schedule (i.e., a complete set of slots), instead of being concerned with a single slot. Thus, we need techniques to decompose the agent utilities over individual slots.

In this section, we first describe (1) how agent utilities are decomposed by resource agents and project agents respectively for making scheduling decisions about a single activity. Then, we present (2) the lease-based slot negotiation scenario that illustrates

how a project agent interacts with one (or more than one) resource agent for scheduling an activity.

4.2.1 Utility Decomposition

In §4.1.1 and §4.1.2, we saw that an agent schedule, both for a resource agent and for a project agent, is composed of a set of slots. Agent utilities are performance measures based on agent schedules. In this subsection, we investigate how resource agents and project agents decompose their utilities over individual slots. We refer to the decomposed utilities as (a) marginal resource-agent utility and (b) marginal project-agent utility, respectively. Below, we derive the two marginal-utility formulas.

A: Marginal Resource-agent Utility

We recall that when a resource agent RA_k considers the resource-levelling objective, the utility $U_{RA_k}(\Pi^k)$ is the negative value of total squared resource utilisation cost (see Equation 4.2).

Let $\pi_{i,j,k}^R$ be the resource-agent slot involving $a_{i,j}$ in the resource-agent schedule Π^k , let $\Pi_{<i,j}^k$ be the *partial schedule* of RA_k when $a_{i,j}$ is going to be scheduled, and let $\Pi_{\leq i,j}^k$ be the partial schedule of RA_k when $a_{i,j}$ is scheduled. Thus, $\Pi_{\leq i,j}^k = \Pi_{<i,j}^k \cup \{\pi_{i,j,k}^R\}$.

The difference between two utilities $U_{RA_k}(\Pi_{\leq i,j}^k)$ and $U_{RA_k}(\Pi_{<i,j}^k)$ is the resource cost of the slot $\pi_{i,j,k}^R$. We refer to the difference as a *marginal resource-agent utility*. We formulate it as follows.

$$U_{RA_k}^{mg}(\Pi_{\leq i,j}^k) = c_k^u \sum_{t \geq 0} [\lambda^2(\Pi_{<i,j}^k, t) - \lambda^2(\Pi_{\leq i,j}^k, t)] \quad (4.10)$$

We recall that $\lambda(\Pi, t)$ is the resource load at the time point t given a schedule Π (see Definition 4.3).

B: Marginal Project-agent Utility

Besides deciding on the decomposition of the resource-agent utility over individual slots, we must also decide how to decompose the project-agent utility over individual activities.

Let $\Pi_{i,j}$ be the schedule of an activity $a_{i,j}$ (see Definition 4.6). $\Pi_{i,j}$ is a set of slots $\{\pi_{i,j,k}^P | k \in \{1, \dots, K\} \wedge r_{i,j}^k \in \mathbb{N}^+\}$ and belongs to the project-agent schedule of PA_i ($\Pi_{i,j} \subset \Pi_i$). Let $\Pi_i^{<i,j}$ be the *partial schedule* of PA_i when activity $a_{i,j}$ is going to be scheduled, and let $\Pi_i^{\leq i,j}$ be the partial schedule of PA_i when $a_{i,j}$ is scheduled. Thus, $\Pi_i^{\leq i,j} = \Pi_i^{<i,j} \cup \Pi_{i,j}$.

We refer to the delay caused by scheduling an activity $a_{i,j}$ as a *marginal delay* $dl^{mg}(\Pi_i^{\leq i,j})$. Equation 4.11 formulates the marginal delay.

$$dl_i^{mg}(\Pi_i^{\leq i,j}) = t_{i,n_i+1}^{es}(\Pi_i^{\leq i,j}) - t_{i,n_i+1}^{es}(\Pi_i^{<i,j}) \quad (4.11)$$

In this marginal delay formulation, the function $t_{i,n_i+1}^{es}(\Pi_i^{\leq i,j})$ denotes the earliest possible start time of the fictitious activity a_{i,n_i+1} given the current schedule $\Pi_i^{\leq i,j}$.

That is, it denotes the *earliest possible completion time* of project P_i given the current schedule $\Pi_i^{\leq i,j}$. Thus the *marginal delay cost* using weight c_i^d becomes $dc_i^{mg}(\Pi_i^{\leq i,j}) = c_i^{dl} \cdot dl_i^{mg}(\Pi_i^{\leq i,j})$.

Now the marginal project-agent utility for scheduling activity $a_{i,j}$ at $s_{i,j}$ is defined as the sum of the marginal delay cost and the marginal resource-agent utility.

$$U_{PA_i}^{mg}(\Pi_i^{\leq i,j}) = c_i^{dl} \cdot dl_i^{mg}(\Pi_i^{\leq i,j}) + U_{RA_k}^{mg}(\Pi_k^{\leq i,j}) \quad (4.12)$$

4.2.2 Lease-based Slot Negotiation

Based on the marginal agent utility functions (see Equation 4.10 and Equation 4.12), we now propose a scenario for inter-agent interactions referred to as a lease-based slot negotiation. The scenario illustrates the interactions between a project agent PA_i and a collection of resource agents $\{RA_k | r_{i,j}^k \in \mathbb{N}^+\}$ for making the schedule of activity $a_{i,j}$. The scenario comprises the following five steps.

Step 1: Sending RfQs (see Figure 4.5) — Project agent PA_i initiates the interaction by sending *requests for quotations* (RfQs) to the corresponding resource agents $\{RA_k | r_{i,j}^k \in \mathbb{N}^+\}$. The RfQ sent to RA_k is a tuple that consists of relevant information² for scheduling $a_{i,j}$:

$$RfQ_{i,j}^k = \langle t_{i,j}^{es}(\Pi_i^{\leq i,j}), t_{i,j}^{ls}(\Pi_i^{\leq i,j}), p_{i,j}, r_{i,j}^k \rangle.$$

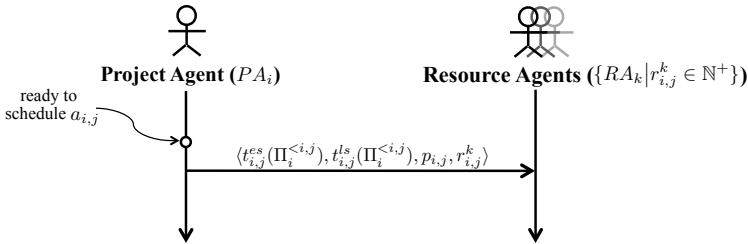


Figure 4.5: Lease-based slot negotiation, step 1 — Sending RfQs

Step 2: Receiving slot offers (see Figure 4.6) — After the receipt of the RfQ sent by PA_i , each RA_k uses its own pricing model (e.g., marginal resource-agent utility function given in Equation 4.10) to calculate a list of slot offers $O_{i,j}^k$, and sends the offers back to PA_i .

²For a problem without a project-deadline constraint, the latest possible start time $t_{i,j}^{ls}(\Pi_i^{\leq i,j})$ of an activity $a_{i,j}$ may be positive infinity: $t_{i,j}^{ls}(\Pi_i^{\leq i,j}) = \infty$. In that case, the project agent will choose an upper bound of the start time for the slot request, denoted by $t_{i,j}^{ub}(\Pi_i^{\leq i,j})$.

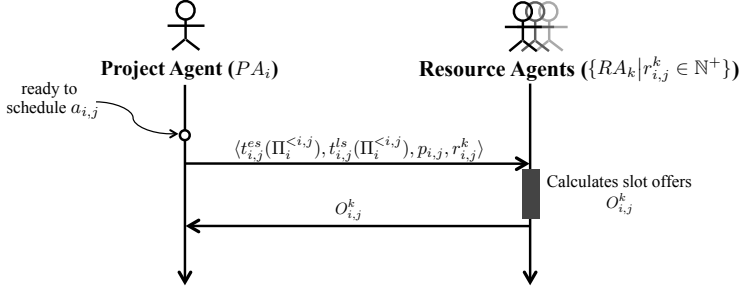


Figure 4.6: Lease-based slot negotiation, step 2 — Receiving slot offers

A slot offer $o_{i,j,l}^k \in O_{i,j}^k$ is a resource-agent slot $\pi_{i,j,k,l}^R$ associated with a price $Pr(\pi_{i,j,k,l}^R)$:

$$o_{i,j,l}^k = \langle \pi_{i,j,k,l}^R, Pr(\pi_{i,j,k,l}^R) \rangle. \quad (4.13)$$

In Equation 4.13, l is the index of the offer $o_{i,j,l}^k$ in the offer list $O_{i,j}^k$.

A	Slot		Price
	I	R	
$a_{i,j}$	$[t_{i,j}^{es}(\Pi_i^{<i,j}), t_{i,j}^{es}(\Pi_i^{<i,j}) + p_{i,j}]$	$(k: r_{i,j}^k)$	$Pr(\pi_{i,j,k,1}^R)$
$a_{i,j}$	$[t_{i,j}^{es}(\Pi_i^{<i,j}) + 1, t_{i,j}^{es}(\Pi_i^{<i,j}) + p_{i,j} + 1]$	$(k: r_{i,j}^k)$	$Pr(\pi_{i,j,k,2}^R)$
$a_{i,j}$	$[t_{i,j}^{es}(\Pi_i^{<i,j}) + 2, t_{i,j}^{es}(\Pi_i^{<i,j}) + p_{i,j} + 2]$	$(k: r_{i,j}^k)$	$Pr(\pi_{i,j,k,3}^R)$
$a_{i,j}$	\dots	$(k: r_{i,j}^k)$	\dots
$a_{i,j}$	$[t_{i,j}^{ls}(\Pi_i^{<i,j}), t_{i,j}^{ls}(\Pi_i^{<i,j}) + p_{i,j}]$	$(k: r_{i,j}^k)$	$Pr(\pi_{i,j,k,x}^R)$

Table 4.1: A list of slot offers $O_{i,j}^k$ for scheduling $a_{i,j}$ from RA_k

Table 4.1 shows a list of offers sent by RA_k to PA_i . The half-open time interval of the earliest slot offer $o_{i,j,1}^k$ is

$$I_{i,j,1}^k = [t_{i,j}^{es}(\Pi_i^{<i,j}), t_{i,j}^{es}(\Pi_i^{<i,j}) + p_{i,j}],$$

and the half-open time interval of the latest slot offer $o_{i,j,x}^k$ is

$$I_{i,j,x}^k = [t_{i,j}^{ls}(\Pi_i^{<i,j}), t_{i,j}^{ls}(\Pi_i^{<i,j}) + p_{i,j}].$$

The resource capacity $c(\pi_{i,j,k,l}^R)$ of the slot $\pi_{i,j,k,l}^R$ in each offer should be equivalent to the resource requirement $r_{i,j}^k$ in the RfQ:

$$c(\pi_{i,j,k,l}^R) = r_{i,j}^k.$$

Step 3: Aggregating slot offers (see Figure 4.7) — PA_i receives from each RA_k a list of slot offers and aggregates the lists into a list of aggregated offers $\hat{O}_{i,j} = \{\hat{o}_{i,j,l}\}$.

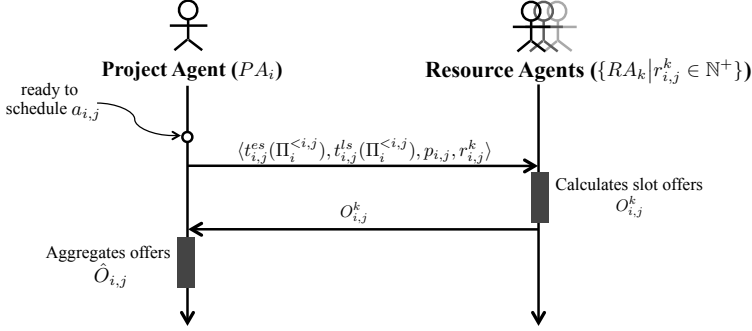


Figure 4.7: Lease-based slot negotiation, step 3 — Aggregating and evaluating slot offers

Each aggregated offer $\hat{o}_{i,j,l}$ contains the slot offers that have the same time interval.

$$\pi_{i,j,k,l'}^R \stackrel{I}{=} \pi_{i,j,k',l''}^R, \quad \forall \pi_{i,j,k,l'}^R, \pi_{i,j,k',l''}^R \in \hat{o}_{i,j,l}$$

The resource price for each aggregated offer $\hat{o}_{i,j,l}$ is the sum of the resource prices of all the slots in $\hat{o}_{i,j,l}$:

$$Pr^R(\hat{o}_{i,j,l}) = \sum_{\pi_{i,j,k,l'}^R \in \hat{o}_{i,j,l}} Pr(\pi_{i,j,k,l'}^R).$$

PA_i adds the potential marginal delay cost (i.e., $dc_i^{mg}(\Pi_{i,l}^{\leq i,j})$, see Equation 4.11) to the summed resource price, and attains a total cost for each aggregated offer:

$$TC(\hat{o}_{i,j,l}) = Pr^R(\hat{o}_{i,j,l}) + dc_i^{mg}(\Pi_{i,l}^{\leq i,j}) = \sum_{\pi_{i,j,k,l'}^R \in \hat{o}_{i,j,l}} Pr(\pi_{i,j,k,l'}^R) + dc_i^{mg}(\Pi_{i,l}^{\leq i,j}).$$

Activity	Aggregated offer		Total cost
	Interval	Resources	
$a_{i,j}$	$[t_{i,j}^{es}(\Pi_i^{<i,j}), t_{i,j}^{es}(\Pi_i^{<i,j}) + p_{i,j})$	$\{(k: r_{i,j}^k)\}$	$\sum_k Pr(\pi_{i,j,k,1}^R) + dc_i^{mg}(\Pi_{i,1}^{\leq i,j})$
$a_{i,j}$	$[t_{i,j}^{es}(\Pi_i^{<i,j}) + 1, t_{i,j}^{es}(\Pi_i^{<i,j}) + p_{i,j} + 1)$	$\{(k: r_{i,j}^k)\}$	$\sum_k Pr(\pi_{i,j,k,2}^R) + dc_i^{mg}(\Pi_{i,2}^{\leq i,j})$
$a_{i,j}$	$[t_{i,j}^{es}(\Pi_i^{<i,j}) + 2, t_{i,j}^{es}(\Pi_i^{<i,j}) + p_{i,j} + 2)$	$\{(k: r_{i,j}^k)\}$	$\sum_k Pr(\pi_{i,j,k,3}^R) + dc_i^{mg}(\Pi_{i,3}^{\leq i,j})$
$a_{i,j}$	\dots	$\{(k: r_{i,j}^k)\}$	\dots
$a_{i,j}$	$[t_{i,j}^{ls}(\Pi_i^{<i,j}), t_{i,j}^{ls}(\Pi_i^{<i,j}) + p_{i,j})$	$\{(k: r_{i,j}^k)\}$	$\sum_k Pr(\pi_{i,j,k,x}^R) + dc_i^{mg}(\Pi_{i,x}^{\leq i,j})$

Table 4.2: Evaluating the aggregated offers for scheduling $a_{i,j}$

Table 4.2 shows a list of aggregated offers and the total cost of each aggregated offer.

Step 4: Sending leases requests (see Figure 4.8) — PA_i sorts the aggregated offers in Table 4.2 based on the total cost, and chooses the aggregated offer with the lowest total cost

$$\arg \min_{\hat{o}_{i,j,l}} TC(\hat{o}_{i,j,l})$$

for scheduling $a_{i,j}$. In case multiple aggregated offers have the same lowest total cost, the one that starts first will be selected. PA_i considers the slots in the chosen aggregated offer as a set of leases, and sends the lease requests to the corresponding resource agents.

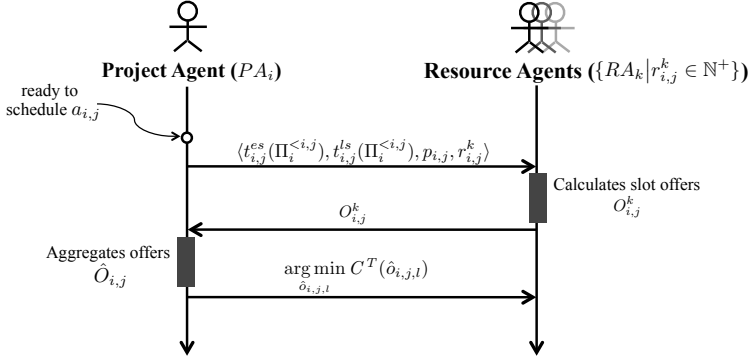


Figure 4.8: Lease-based slot negotiation, step 4 — Sending leases requests

Step 5: Making leases (see Figure 4.9) — Resource agent RA_k in $\{RA_k | r_{i,j}^k \in \mathbb{N}^+\}$ receiving a lease request $(\pi_{i,j,k,l}^R)$ adds the lease to its schedule $(\pi_{i,j,k,l}^R \in \Pi^k)$, and sends an acknowledgement message to RA_i . After receiving the message, PA_i adds an equivalent slot $\pi_{i,j,k,l}^P = \pi_{i,j,k,l}^R$ to its agent schedule $(\pi_{i,j,k,l}^P \in \Pi_i)$.

Once making the schedule of $a_{i,j}$ has been completed, PA_i can move on to schedule the next unscheduled activity.

We note that in step 1 of the negotiation scenario, the RfQs sent by the project agent PA_i specify the earliest and latest possible start times of an activity. This guarantees that the offers received in Step 2, as well as the leases made in Step 5 will neither violate the precedence constraints nor the additional temporal constraint.

Likewise, in step 2 of the negotiation scenario, in case a slot $\pi_{i,j,k,l}^R$ violates the resource-capacity constraint of RA_k , the price of the slot $Pr(\pi_{i,j,k,l}^R)$ will be set to be (positive) infinity by RA_k . The underlying idea is that the project agent PA_i will not lease a slot with an infinite cost. In this way, resource-capacity constraints are not violated.

As long as the leases chosen makes both the resource-agent schedules and the project-agent schedule feasible, a global feasible schedule can be obtained.

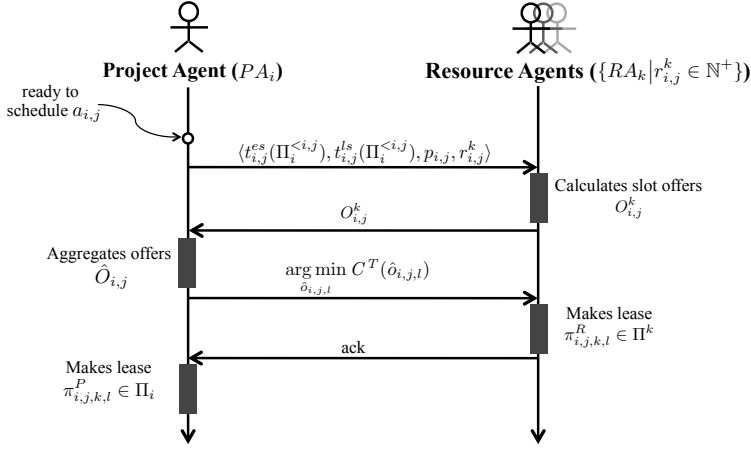
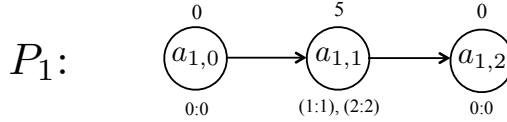


Figure 4.9: Lease-based slot negotiation, step 5 — Making leases

An Example

Below, we illustrate the lease-based slot negotiation scenario by an example. In the example, a project P_1 in a DRCMPSP is released at $\bar{r}l_1 = 0$. The project has a due-time constraint $\bar{dt}_1 = 8$, and a deadline constraint $\bar{dl}_1 = 15$. P_1 comprises only one real activity $a_{1,1}$, of which the mode $\mu_{1,1} = \langle \{(1:1), (2:2)\}, 5 \rangle$. The AoN network of P_1 is depicted in Figure 4.10.

Figure 4.10: AoN network of an example project P_1

The mode $\mu_{1,1}$ specifies that (1) processing activity $a_{1,1}$ requires one unit of resource type R_1 and two units of resource type R_2 , (2) the estimated processing time is five time units.

Scheduling $a_{1,1}$ using the proposed lease-based slot negotiation scenario requires interactions among three agents — one project agent (PA_1) and two resource agents (RA_1 and RA_2). Below, we describe the scenario.

In step 1, PA_1 sends out two RfQs: (i) $RfQ_{1,1}^1 = \langle 0, 10, 5, 1 \rangle$ to RA_1 , and (ii) $RfQ_{1,1}^2 = \langle 0, 10, 5, 2 \rangle$ to RA_2 . In step 2, PA_1 receives from each of the two resource agents a list of slot offers (see Table 4.3 and Table 4.4). The calculations of the offer prices are carried out by the two resource agents autonomously based on their own (resource-agent) utilities. In Step 3, PA_1 aggregates the two lists of offers and calculates the total cost of each

aggregated offer³. We can see from Table 4.5 that the aggregated offer No. 4 has the lowest total cost. Accordingly, in step 4, PA_1 requests two leases for scheduling $a_{1,1}$: $\pi_{1,1,1}^R = \langle a_{1,1}, [3, 8), (1: 1) \rangle$ and $\pi_{1,1,2}^R = \langle a_{1,1}, [3, 8), (2: 2) \rangle$. In step 5, RA_1 adds $\pi_{1,1,1}^R$ to schedule Π^1 : $\pi_{1,1,1}^R \in \Pi^1$; similarly, RA_2 adds $\pi_{1,1,2}^R$ to schedule Π^2 : $\pi_{1,1,2}^R \in \Pi^2$. After receiving the acknowledgements from both of the two resource agents RA_1 and RA_2 , PA_1 adds the two slots $\pi_{1,1,1}^P$ and $\pi_{1,1,2}^P$ to schedule Π_1 resulting in $\{\pi_{1,1,1}^P, \pi_{1,1,2}^P\} \subset \Pi_1$. The two slots together constitute by definition the schedule $\Pi_{1,1}$ of $a_{1,1}$: $\Pi_{1,1} = \{\pi_{1,1,1}^P, \pi_{1,1,2}^P\}$ (See Figure 4.11).

Offer	Slot			Price
	A	I	R	
1	$a_{1,1}$	[0, 5)	(1: 1)	300
2	$a_{1,1}$	[1, 6)	(1: 1)	200
3	$a_{1,1}$	[2, 7)	(1: 1)	150
4	$a_{1,1}$	[3, 8)	(1: 1)	150
5	$a_{1,1}$	[4, 9)	(1: 1)	140
6	$a_{1,1}$	[5, 10)	(1: 1)	120
7	$a_{1,1}$	[6, 11)	(1: 1)	120
8	$a_{1,1}$	[7, 12)	(1: 1)	100
9	$a_{1,1}$	[8, 13)	(1: 1)	100
10	$a_{1,1}$	[9, 14)	(1: 1)	100
11	$a_{1,1}$	[10, 15)	(1: 1)	100

Table 4.3: List of slot offers sent by RA_1

Offer	Slot			Price
	A	I	R	
1	$a_{1,1}$	[0, 5)	(2: 2)	400
2	$a_{1,1}$	[1, 6)	(2: 2)	350
3	$a_{1,1}$	[2, 7)	(2: 2)	350
4	$a_{1,1}$	[3, 8)	(2: 2)	320
5	$a_{1,1}$	[4, 9)	(2: 2)	320
6	$a_{1,1}$	[5, 10)	(2: 2)	300
7	$a_{1,1}$	[6, 11)	(2: 2)	270
8	$a_{1,1}$	[7, 12)	(2: 2)	250
9	$a_{1,1}$	[8, 13)	(2: 2)	200
10	$a_{1,1}$	[9, 14)	(2: 2)	200
11	$a_{1,1}$	[10, 15)	(2: 2)	200

Table 4.4: List of slot offers sent by RA_2

Aggregated Offer	Activity	Interval	Resources	Marginal Delay Cost	Resource Costs	Total Cost
1	$a_{1,1}$	[0, 5)	$\{(1: 1), (2: 2)\}$	0	700	700
2	$a_{1,1}$	[1, 6)	$\{(1: 1), (2: 2)\}$	0	550	550
3	$a_{1,1}$	[2, 7)	$\{(1: 1), (2: 2)\}$	0	500	500
4	$a_{1,1}$	[3, 8)	$\{(1: 1), (2: 2)\}$	0	470	470
5	$a_{1,1}$	[4, 9)	$\{(1: 1), (2: 2)\}$	100	460	560
6	$a_{1,1}$	[5, 10)	$\{(1: 1), (2: 2)\}$	200	420	620
7	$a_{1,1}$	[6, 11)	$\{(1: 1), (2: 2)\}$	300	390	690
8	$a_{1,1}$	[7, 12)	$\{(1: 1), (2: 2)\}$	400	350	750
9	$a_{1,1}$	[8, 13)	$\{(1: 1), (2: 2)\}$	500	300	800
10	$a_{1,1}$	[9, 14)	$\{(1: 1), (2: 2)\}$	600	300	900
11	$a_{1,1}$	[10, 15)	$\{(1: 1), (2: 2)\}$	700	300	1000

Table 4.5: Aggregated offers for scheduling $a_{1,1}$

With the scenario of the interaction between a project agent and two resource agents as illustrated above, we successfully distribute the decision-making responsibilities and

³We assume that the unit-time delay cost of P_1 is $c_1^d = 100$.

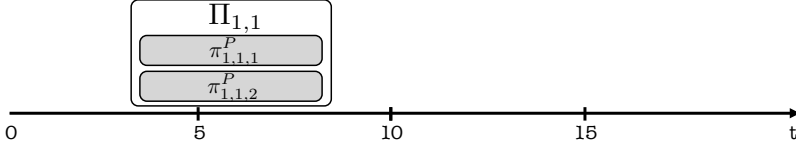


Figure 4.11: Schedule of $a_{i,j}$ on the timeline of PA_1

concerns among the individual and different types of agents. As a result of the lease-based slot negotiation scenario, the overall schedule emerges.

4.3 Answer to Research Question 1

In this chapter, we addressed the first research question (see Section 1.2). [RQ1: *How can an AGH scheduling problem be represented in an agent-based model?*] We showed that the AGH scheduling problem can be effectively modelled by an agent-based model. The proposed model consists of (1) two classes of role-based agents — resource agents and project agents — and (2) a lease-based market mechanism for coordinating the autonomous scheduling decisions of the individual agents.

Based on the illustrations of our proposed model in this chapter, we may answer the first research question as follows. The essence of agent-based modelling lies in two different aspects: (1) agent representation and (2) agent interaction. Both are briefly discussed below.

Agent representation

For the first modelling aspect, we recall that in Section 2.2 an AGH scheduling problem is formulated as an instance of a DRCMPSP/u. A DRCMPSP/u concerns two classes of entities/organisations — resource-type managers and project managers. The managers may have their own self-interested objectives. Thus, in Section 4.1, we adopted a physical-entity-oriented agent-modelling approach, and modelled these two classes of entities as two classes of agents: resource agents and project agents, respectively. The chosen physical-entity-oriented modelling approach provides a natural description of the AGH domain by incorporating the two “behavioural” entities in a DRCMPSP/u. For modelling project agents, we chose to use a ‘coarse-grained’ approach. The chosen ‘coarse-grained’ approach allows the modelling of self-interested agents and avoids inter-agent communication overloads. The self-interested nature of the agents is represented by the utility modelling of each of the two classes of agents. The chosen agent representation offers properties such as self-interestedness and scalability to the agent-based scheduling system.

Agent interaction

For the second modelling aspect the following holds: in order to realise agent interaction, a common “language” has to be defined. In the proposed model, we introduced a concept of resource-time slot, that is used in both the resource-agent schedule and the project-agent

schedule. In Section 4.2, we proposed a lease-based market mechanism. The mechanism coordinates the scheduling decisions among two classes of heterogeneous agents. For processing an activity, a lease (i.e., a time-resource slot) is negotiated by a resource agent and a project agent. The agents evaluate the value of the slot based on their own value systems (marginal agent utilities). As a result, the proposed coordination mechanism successfully distributes the scheduling decisions over autonomous decision makers, and as long as the slot chosen makes both the resource-agent schedule and the project-agent schedule feasible, a global feasible schedule will be obtained. The proposed coordination mechanism offers properties such as openness and efficiency to the agent-based scheduling system.

Chapter 5

Online Iterative Scheduling

In a partially observable project-scheduling environment, pre-determined project schedules, i.e., schedules made before all projects have been released, frequently have to be revised. This happens rather often in an AGH-scheduling environment where aircraft often cannot arrive exactly at their expected arrival times. As discussed in §2.2.3, partial observability in a project-scheduling context is presented as variable project release times. In this chapter, we address problems characterised by the first class of uncertainty. Accordingly, we will answer research question 2, which reads as follows.

RQ2: *How can agents make and coordinate their local decisions in order to achieve a globally efficient and robust schedule in a partially observable environment?*

Scheduling solutions to problems that are under the nondeterministic aspect of uncertainty will be further investigated in Chapter 6 when we answer RQ3.

A straightforward solution to problems with variable project release times would be to schedule the projects *online* when the projects are actually released. Online scheduling may guarantee the robustness of the schedules against variable project release times. However, it can simultaneously be highly inefficient because the projects are scheduled sequentially according to the order of their actual release times. In this chapter, we propose an *online iterative* (OI) approach (denoted by OI-MAS) in the proposed MAS scheduling framework. OI-MAS aims at constructing an efficient and robust AGH schedule under variable project release times.

OI-MAS consists of two components: (1) a *clairvoyant online schedule generation scheme* (COSGS) and (2) an *iterative schedule-improvement method* (ISIM). The details of the two components will be presented in Section 5.1 and in Section 5.2, respectively. In Section 5.3, we conduct experiments to evaluate the performance of the proposed approach, and provide empirical analyses of the system performance compared to three state-of-art centralised and decentralised approaches (SASP, Bwd/Fwd, and GT-MAS). Section 5.4 concludes this chapter by answering RQ2.

5.1 Clairvoyant Online Schedule Generation Scheme

In this section, we design a scheduling scheme that guides the scheduling procedure in the proposed MAS scheduling system for solving DRCMPSP/u, in which variable project release times occur. The designed scheme is in two levels. First, on the project level, we adopt a clairvoyant online scheme (COS) in which the scheduling procedure of a project can only start when the project is actually released. Second, on the activity level, we adopt a *serial schedule generation scheme* (s-SGS) in which the schedule of a project is built stepwise in accordance with the activity precedence relations. Below, we discuss these two schemes in detail.

5.1.1 Clairvoyant Online Scheme

Traditional (offline) scheduling consists of employing a scheduling scheme where the scheduling procedure can be carried out with full knowledge of the problem instance. The knowledge is available in advance. However, in most planning and scheduling problems it is unlikely that all information necessary to define a problem instance is available in advance. In an AGH scheduling environment, the relevant project information is gradually revealed along with the releases of the projects over time. As stated above, here we focus on problems with variable project release times. For projects having expected release times, the **variable** project release times mean that the actual project release times are different from the expected ones. The variable project release times can also mean that unanticipated new projects need to be incorporated on the fly.

In the context of variable project release times, two online scheduling schemes have been proposed in the literature (cf. Vestjens, 1997): (1) a *clairvoyant* scheme and (2) a *non-clairvoyant* scheme. If the online scheduling is clairvoyant, then the processing requirement of all activities of a project is known as soon as the project is released or even earlier. If the online scheduling is non-clairvoyant, then the processing requirements of all activities of a project are not completely known at the project's release time. Instead, the information is gradually revealed along with the progress of the project. In the latter case, the project managers may, in the worst case, have to wait with scheduling an activity until all the preceding activities have finished.

In the AGH-scheduling environment, the time scale on which services are required by the aircraft makes the non-clairvoyant scheme impractical. The ground-service providers need some preparation time before the actual services are to be provided. For this reason, we adopt the clairvoyant online scheme (COS) in our MAS scheduling framework, i.e., assuming that the information of all activities of a project becomes known at the latest at the moment the project is released. In practice, the starting activities (i.e., activities having no (real) predecessors) of a project also require a certain amount of preparation time before they can actually be performed. Therefore, a clairvoyant scheme assumes that the information about starting activities of a project are known some time earlier than the actual release time of a project, giving sufficient time for the preparation.

The COS employed is an incremental scheme in which the order of the projects to be scheduled in a DRCMPSP/u is determined according to the order of the (actual) project release times \overline{rl}_i^* . In the next subsection, we discuss the scheme on the activity level,

that is a scheme employed by each of the individual project agents to scheduling all its activities.

5.1.2 Schedule Generation Schemes

Schedule generation schemes (SGSs) are the core of most centralised heuristic solution procedures for RCPSP (cf. Kolisch and Hartmann, 1999). SGSs start from scratch and build a feasible project schedule by stepwise extension of a partial project schedule. Two different SGSs are distinguished: a *serial* SGS (s-SGS) and a *parallel* SGS (p-SGS). In both s-SGS and p-SGS, a schedule of a project is constructed incrementally, where s-SGS performs an activity-incremental procedure and p-SGS performs a time-incremental procedure. A project agent at each step in both schemes chooses only one activity at a time from a set of (precedent) eligible activities (i.e., non-scheduled activities of which the predecessors are scheduled) to schedule. In case multiple activities are eligible, the decision of choosing which activity is generally made through a priority-rule heuristic (see §3.1.2).

Readers who are interested in the details of these two schemes are referred to the work by Kolisch and Hartmann (1999) for further reading. In the thesis, we adopt the s-SGS for intra-project scheduling.

Combining COS and SGS, we have a complete scheme for scheduling a DRCMPSP/u — the *clairvoyant online schedule generation scheme* (COSGS).

5.1.3 An Example

Below, we use an example project P_1 in a DRCMPSP/u to illustrate the COSGS. The project P_1 has an expected release time $\bar{r}l_1 = 0$, and a due-time constraint $\bar{d}t_1 = 16$. P_1 consists of three precedence-related real activities: $a_{1,1} \prec a_{1,2} \prec a_{1,3}$. The AoN network of P_1 is depicted in Figure 5.1.

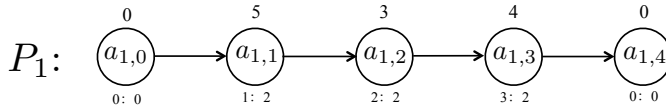


Figure 5.1: AoN network of an example project P_1

According to Figure 5.1, the processing modes of the three real activities are as follows.

$$\mu_{1,1} = \langle \{(1: 2)\}, 5 \rangle$$

$$\mu_{1,2} = \langle \{(2: 2)\}, 3 \rangle$$

$$\mu_{1,3} = \langle \{(3: 2)\}, 4 \rangle$$

In the context of variable project release times, we assume that P_1 is released 1 time unit later than the expected release time ($\bar{r}l_1^* = 1$). In the COSGS, the clairvoyant online scheme prevents this late-releasing incident from invalidating the current schedule.

	Slot			Price
	A	I	R	
1	$a_{1,1}$	[1, 6]	(1: 2)	600
2	$a_{1,1}$	[2, 7]	(1: 2)	600
3	$a_{1,1}$	[3, 8]	(1: 2)	600
4	$a_{1,1}$	[4, 9]	(1: 2)	600
5	$a_{1,1}$	[5, 10]	(1: 2)	600
6	$a_{1,1}$	[6, 11]	(1: 2)	600
7	$a_{1,1}$	[7, 12]	(1: 2)	600
8	$a_{1,1}$	[8, 13]	(1: 2)	600
\vdots	\vdots	\vdots	\vdots	\vdots

Table 5.1: Offers sent by RA_1

	Aggregated offer			Marginal Delay Cost	Resource Costs	Total Cost
	A	I	R_s			
1	$a_{1,1}$	[1, 6]	$\{(1: 2)\}$	0	600	600
2	$a_{1,1}$	[2, 7]	$\{(1: 2)\}$	0	600	600
3	$a_{1,1}$	[3, 8]	$\{(1: 2)\}$	0	600	600
4	$a_{1,1}$	[4, 9]	$\{(1: 2)\}$	0	600	600
5	$a_{1,1}$	[5, 10]	$\{(1: 2)\}$	100	600	700
6	$a_{1,1}$	[6, 11]	$\{(1: 2)\}$	200	600	800
7	$a_{1,1}$	[7, 12]	$\{(1: 2)\}$	300	600	900
8	$a_{1,1}$	[8, 13]	$\{(1: 2)\}$	400	600	1000
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots

Table 5.2: Aggregated offer for scheduling $a_{1,1}$

	Slot			Price
	A	I	R	
1	$a_{1,2}$	[6, 9]	(2: 2)	600
2	$a_{1,2}$	[7, 10]	(2: 2)	550
3	$a_{1,2}$	[8, 11]	(2: 2)	500
4	$a_{1,2}$	[9, 12]	(2: 2)	500
5	$a_{1,2}$	[10, 13]	(2: 2)	400
6	$a_{1,2}$	[11, 14]	(2: 2)	400
7	$a_{1,2}$	[12, 15]	(2: 2)	300
8	$a_{1,2}$	[13, 16]	(2: 2)	300
\vdots	\vdots	\vdots	\vdots	\vdots

Table 5.3: Offers sent by RA_2

	Aggregated offer			Marginal Delay Cost	Resource Costs	Total Cost
	A	I	R_s			
1	$a_{1,2}$	[6, 9]	$\{(2: 2)\}$	0	600	600
2	$a_{1,2}$	[7, 10]	$\{(2: 2)\}$	0	550	550
3	$a_{1,2}$	[8, 11]	$\{(2: 2)\}$	0	500	500
4	$a_{1,2}$	[9, 12]	$\{(2: 2)\}$	0	500	500
5	$a_{1,2}$	[10, 13]	$\{(2: 2)\}$	100	400	500
6	$a_{1,2}$	[11, 14]	$\{(2: 2)\}$	200	400	600
7	$a_{1,2}$	[12, 15]	$\{(2: 2)\}$	300	300	600
8	$a_{1,2}$	[13, 16]	$\{(2: 2)\}$	400	300	700
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots

Table 5.4: Aggregated offers for scheduling $a_{1,2}$

Instead, PA_1 starts to make its project-agent schedule as soon as P_1 is **actually** released. The scheduling process of PA_1 is carried out in an s-SGS, in which the three activities are scheduled sequentially.

The proposed lease-base market mechanism (see Section 4.2) is employed to schedule each of the three activities. For instance, P_1 starts by negotiating with the resource agent¹ RA_1 for making a schedule of $a_{1,1}$. The slot offers sent by RA_1 are listed in Table 5.1. The aggregated offers and the total cost² of each aggregated offer can be found in Table 5.2. In the table, we can see that four aggregated offers (No. 1 to No. 4) have the same lowest total cost (600). The first offer that has a slot $\langle [1, 6), (1: 2), a_{1,1} \rangle$ is chosen by PA_1 to schedule $a_{1,1}$ since among four offers it is the first that starts (see step 4 in the slot negotiation scenario in §4.2.2). The schedule of the activity $a_{1,1}$ is decided: $\Pi_{1,1} = \{\pi_{1,1,1}^P\} = \{\langle [1, 6), (1: 2), a_{1,1} \rangle\}$.

Similarly, $a_{1,2}$ and $a_{1,3}$ are scheduled using the same negotiation scenario. Table 5.3 to 5.6 show the slot offers and the aggregated offers for scheduling $a_{1,2}$ and $a_{1,3}$. Eventually,

¹In the example, each of the three activities requires only one resource type (R_1 for $a_{1,1}$, R_2 for $a_{1,2}$, and R_3 for $a_{1,3}$, respectively). Thus, P_1 negotiates with only one resource agent for scheduling an activity. In practice, a project agent might need to negotiate with more than one resource agent in case an activity requires multiple resource types.

²We assume the delay cost per unit time of P_1 is $c_1^{dl} = 100$.

	Slot			Price
	<i>A</i>	<i>I</i>	<i>R</i>	
1	$a_{1,3}$	[11, 15]	(3: 2)	900
2	$a_{1,3}$	[12, 16]	(3: 2)	900
3	$a_{1,3}$	[13, 17]	(3: 2)	700
4	$a_{1,3}$	[14, 18]	(3: 2)	700
5	$a_{1,3}$	[15, 19]	(3: 2)	400
6	$a_{1,3}$	[16, 20]	(3: 2)	400
7	$a_{1,3}$	[17, 21]	(3: 2)	400
8	$a_{1,3}$	[18, 22]	(3: 2)	400
⋮	⋮	⋮	⋮	⋮

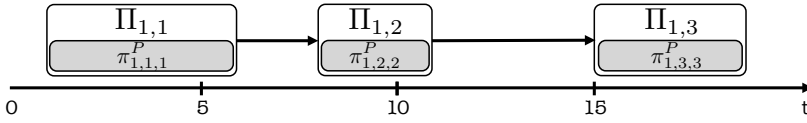
Table 5.5: Offers sent by RA_3

	Aggregated offer			Marginal Delay Cost	Resource Costs	Total Cost
	<i>A</i>	<i>I</i>	<i>Rs</i>			
1	$a_{1,3}$	[11, 15]	$\{(3: 2)\}$	0	900	900
2	$a_{1,3}$	[12, 16]	$\{(3: 2)\}$	0	900	900
3	$a_{1,3}$	[13, 17]	$\{(3: 2)\}$	100	700	800
4	$a_{1,3}$	[14, 18]	$\{(3: 2)\}$	200	700	900
5	$a_{1,3}$	[15, 19]	$\{(3: 2)\}$	300	400	700
6	$a_{1,3}$	[16, 20]	$\{(3: 2)\}$	400	400	800
7	$a_{1,3}$	[17, 21]	$\{(3: 2)\}$	500	400	900
8	$a_{1,3}$	[18, 22]	$\{(3: 2)\}$	600	400	1000
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 5.6: Aggregated offers for scheduling $a_{1,3}$

a project-agent schedule Π_1 is obtained. Figure 5.2 depicts the project-agent schedule on the project's timeline.

$$\begin{aligned}
\Pi_1 &= \{\Pi_{1,1}, \Pi_{1,2}, \Pi_{1,3}\} = \{\{\pi_{1,1,1}^P\}, \{\pi_{1,2,2}^P\}, \{\pi_{1,3,3}^P\}\} \\
&= \{([1, 6), (1: 2), a_{1,1}), ([8, 11), (2: 2), a_{1,2}), ([15, 19), (3: 2), a_{1,3})\}
\end{aligned} \tag{5.1}$$

Figure 5.2: Project-agent schedule Π_1 made by the COSGS

The total cost of the project-agent schedule Π_1 consists of two parts: (1) a project delay cost (i.e., 300), and (2) a total resource cost (i.e., $600 + 500 + 400 = 1500$). In all, the total cost of Π_1 is 1800. Accordingly, the utility of PA_1 is -1800 .

5.1.4 A Discussion of Employing COSGS

We note that the choice of employing the COSGS completely overcomes the schedule disruption caused by variable project-release-time uncertainty. However, there are potential drawbacks. We mention (1) scheduling online reduces the amount of available information, (2) scheduling online can result in losing opportunities for a better allocation of the resources to the activities, and (3) both COS and SGS are one-pass scheduling schemes without backtracking. The latter drawback implies that earlier determined activity schedules have no further changes in the COSGS. In order to reduce these drawbacks, we propose in the following section a method to improve iteratively the earlier obtained schedule in the course of scheduling.

5.2 Iterative Schedule-improvement Method

In this section, we describe an *iterative schedule-improvement method* (ISIM). For the sake of providing utmost clarity, we emphasise that **iterative** and **incremental** are in no way synonyms. In very special cases they are, but certainly not in our OI-MAS approach to DRCMPSP/u.

In OI-MAS, we use two incremental schemes (COS and sSGS) and one iterative scheme (ISIM). The COS is an incremental scheme on the project level and the sSGS is an incremental scheme on the activity level. A complete schedule of a DRCMPSP/u is first constructed project by project, and activity by activity in the COSGS. Afterwards, in the ISIM a project-agent schedule is iteratively reconsidered for improvement.

It is important to notice that agent-based modelling enables autonomous decision making. This allows us to propose a schedule-improvement method in which agents can continuously search for opportunities from which they can improve the current schedules. Schedule improvement is meant in terms of increasing an agent's utility. We note that there are two cases in which an activity schedule can be improved: (1) when there is a schedule change of one of the activity's neighbouring activities (i.e., the activity's immediate predecessors and immediate successors), and (2) when there is a change of the resource-type profile that is used by the activity. In the sequel, we describe in detail how the ISIM improves the earlier obtained schedules in these two cases (§5.2.1 and §5.2.2), respectively.

5.2.1 ISIM by Secure-time-window Update

We recall that an activity $a_{i,j}$ has a dynamic earliest possible start time $t_{i,j}^{es}$ and a dynamic latest possible start time $t_{i,j}^{ls}$ in the scope of the scheduling process. The earliest possible start time $t_{i,j}^{es}$ of $a_{i,j}$ can be computed using Equation 5.2.

$$t_{i,j}^{es} = \max(\overleftarrow{rl}_i^*, s_{i,l} + p_{i,l}, t_{i,o}^{es} + p_{i,o}) \quad (5.2)$$

Equation 5.2 can be interpreted as follows. If $a_{i,j}$ has no immediate predecessors ($\overleftarrow{A}_{i,j} = \emptyset$), the earliest possible start time $t_{i,j}^{es}$ of $a_{i,j}$ equals to the actual release time \overleftarrow{rl}_i^* of the project P_i . Otherwise, if $\overleftarrow{A}_{i,j} \neq \emptyset$ and $a_{i,l}$ is a **scheduled** immediate predecessor of $a_{i,j}$ ($a_{i,l} \prec a_{i,j} \wedge \Pi_{i,l} \neq \emptyset$), $t_{i,j}^{es}$ should not be smaller than the scheduled finish time ($s_{i,l} + p_{i,l}$) of $a_{i,l}$. Furthermore, if $\overleftarrow{A}_{i,j} \neq \emptyset$ and $a_{i,o}$ is an **unscheduled** immediate predecessor of $a_{i,j}$ ($a_{i,o} \prec a_{i,j} \wedge \Pi_{i,o} = \emptyset$), $t_{i,j}^{es}$ should be no smaller than the earliest possible finish time ($t_{i,o}^{ef} = t_{i,o}^{es} + p_{i,o}$) of $a_{i,o}$.

Similarly, the latest possible start time $t_{i,j}^{ls}$ of $a_{i,j}$ can be computed using Equation 5.3.

$$t_{i,j}^{ls} = \min(\overrightarrow{dl}_i - p_{i,j}, s_{i,l} - p_{i,j}, t_{i,o}^{ls} - p_{i,j}) \quad (5.3)$$

While scheduling the activity $a_{i,j}$, project agent PA_i communicates these two time points ($t_{i,j}^{es}$ and $t_{i,j}^{ls}$) to the corresponding resource agents for the enquiry of slot offers (see step 1 in the slot negotiation in §4.2.2). Since $a_{i,j}$ can only start in between the two time points mentioned (both inclusive), we arrive at Equation 5.4.

$$t_{i,j}^{es} \leq s_{i,j} \leq t_{i,j}^{ls} \quad (5.4)$$

Once all the neighbouring activities (immediate predecessors and immediate successors) of an activity $a_{i,j}$ are scheduled, rescheduling $a_{i,j}$ within the updated earliest/latest possible start time will not cause any further delay of P_i . We therefore define a *secure time window* for an activity when all the neighbouring activities of the activity are scheduled.

DEFINITION 5.1 Secure Time Window. *The secure time window of an activity $a_{i,j}$ is a time interval that exists only when all the neighbouring activities of $a_{i,j}$ are scheduled. The time interval starts from the activity's earliest possible start time $t_{i,j}^{es}$ (inclusive), and ends by the activity's latest possible finish time $t_{i,j}^{lf}$ (exclusive). Formally, we have*

$$I_{i,j}^s = [t_{i,j}^{es}, t_{i,j}^{lf}).$$

In the following, we describe how to use the secure time window of an activity to improve the activity's current schedule.

We illustrate the ISIM by using the same example project P_1 as in §5.1.3. We consider the project schedule Π_1 (Equation 5.1) obtained by the COSGS as the initial schedule. In general, ISIM strives for improving Π_i in terms of the project-agent utility $U_{PA_i}(\Pi_i)$.

Below, we describe the first iteration of the ISIM. Without loss of generality, we assume that the resource prices of the slot offers sent by the resource agents in the first iteration are the same as they were in the COSGS.

We first look at the activity $a_{1,1}$. The secure time window $I_{1,1}^s = [1, 8)$ of $a_{1,1}$ was “activated” by the scheduling of the activity $a_{1,2}$ in the COSGS (see Figure 5.3). The secure time window $I_{1,1}^s$ allows PA_1 to send new RfQs in the ISIM. In consequence, three slot offers are received from RA_1 (see Table 5.7). From the aggregated offer list in Table 5.8, we can see that the schedule $\pi_{1,1,1}^P = \langle [1, 6), (1: 2), a_{1,1} \rangle$ remains the best option among all three slots. Therefore, no improvement can be made with respect to $\Pi_{1,1}$ in the first iteration of ISIM.

	Slot			Price
	A	I	R	
1	$a_{1,1}$	[1, 6)	(1: 2)	600
2	$a_{1,1}$	[2, 7)	(1: 2)	600
3	$a_{1,1}$	[3, 8)	(1: 2)	600

Table 5.7: Slot offers sent by RA_1 in the first iteration of the ISIM

	Aggregated offer			Marginal Delay Cost	Resource Costs	Total Cost
	A	I	Rs			
1	$a_{1,1}$	[1, 6)	{(1: 2)}	0	600	600
2	$a_{1,1}$	[2, 7)	{(1: 2)}	0	600	600
3	$a_{1,1}$	[3, 8)	{(1: 2)}	0	600	600

Table 5.8: Aggregated offers for scheduling $a_{1,1}$ in the first iteration of the ISIM

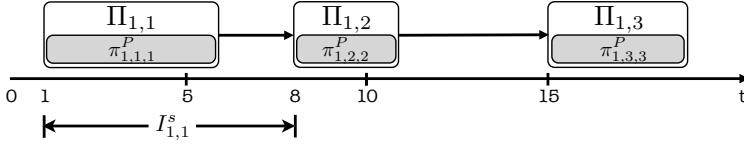


Figure 5.3: The secure time window $I_{1,1}^s$ of $a_{1,1}$ in iteration 1

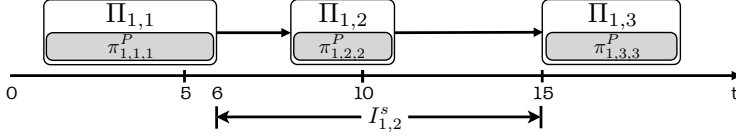


Figure 5.4: The secure time window $I_{1,2}^s$ of $a_{1,2}$ in iteration 1

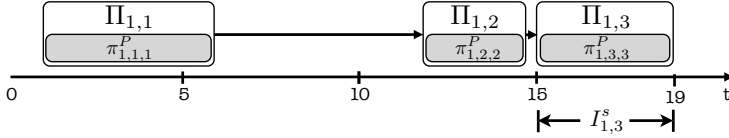


Figure 5.5: The secure time window $I_{1,3}^s$ of $a_{1,3}$ in iteration 1

For the activity $a_{1,2}$, a secure time window $I_{1,2}^s = [6, 15]$ was “activated” by the scheduling of $a_{1,3}$ in the COSGS (see Figure 5.4). In consequence, seven slot offers are received from RA_2 (see Table 5.9). According to Table 5.10, a better schedule for $a_{1,2}$ can be made: $\pi_{1,2,2}^P = \langle [12, 15], (2: 2), a_{1,2} \rangle$. The change of the schedule $\pi_{1,2,2}^P$ from $\langle [8, 11], (2: 2), a_{1,2} \rangle$ to $\langle [12, 15], (2: 2), a_{1,2} \rangle$ will not cause any further delay of P_1 . Instead, it gives a lower resource cost (300) compared to the resource cost (500) of $\pi_{1,2,2}^P$ in the COSGS.

Once the new schedule of $a_{1,2}$ is determined, the secure time window of $a_{1,3}$ has been changed to $I_{1,3}^s = [15, 19]$ (see Figure 5.5). The secure time window coincides with the schedule of $a_{1,3}$. Evidently, no schedule improvement can be made for $a_{1,3}$ in iteration 1.

In all, PA_1 through the first iteration of the ISIM revises the schedule $\Pi_{1,2}$ of $a_{1,2}$ from $\langle [8, 11], (2: 2), a_{1,2} \rangle$ to $\langle [12, 15], (2: 2), a_{1,2} \rangle$. The schedule revision results an improvement of total cost reduction of 200. The total cost of the project-agent schedule of PA_1 in iteration 2 is 1600, and the utility of PA_1 is -1600 .

$$\begin{aligned}\Pi_{1,1} &= \{\pi_{1,1,1}^P\} = \{\langle [0, 5], \{(1: 2)\}, a_{1,1} \rangle\} \\ \Pi_{1,2} &= \{\pi_{1,2,2}^P\} = \{\langle [12, 15], \{(2: 2)\}, a_{1,2} \rangle\} \\ \Pi_{1,3} &= \{\pi_{1,3,3}^P\} = \{\langle [15, 19], \{(3: 2)\}, a_{1,3} \rangle\}\end{aligned}$$

Likewise, the schedule update of $\Pi_{1,2}$ causes a new update of the secure time window $I_{1,1}^s$ of the activity $a_{1,1}$. This creates a new opportunity for PA_1 to reiterate its schedule (i.e., iteration 2) for improvement. As a result, it turns out that the efficiency of Π_1 cannot

	Slot			Price
	A	I	R	
1	$a_{1,2}$	[6, 9]	(2: 2)	600
2	$a_{1,2}$	[7, 10]	(2: 2)	550
3	$a_{1,2}$	[8, 11]	(2: 2)	500
4	$a_{1,2}$	[9, 12]	(2: 2)	500
5	$a_{1,2}$	[10, 13]	(2: 2)	400
6	$a_{1,2}$	[11, 14]	(2: 2)	400
7	$a_{1,2}$	[12, 15]	(2: 2)	300

Table 5.9: Slot offers by RA_2 in the first iteration of the ISIM

	Aggregated offer			Marginal Delay Cost	Resource Costs	Total Cost
	A	I	Rs			
1	$a_{1,2}$	[6, 9]	$\{(2: 2)\}$	0	600	600
2	$a_{1,2}$	[7, 10]	$\{(2: 2)\}$	0	550	550
3	$a_{1,2}$	[8, 11]	$\{(2: 2)\}$	0	500	500
4	$a_{1,2}$	[9, 12]	$\{(2: 2)\}$	0	500	500
5	$a_{1,2}$	[10, 13]	$\{(2: 2)\}$	0	400	400
6	$a_{1,2}$	[11, 14]	$\{(2: 2)\}$	0	400	400
7	$a_{1,2}$	[12, 15]	$\{(2: 2)\}$	0	300	300

Table 5.10: Aggregated offers for scheduling $a_{1,2}$ in the first iteration of the ISIM

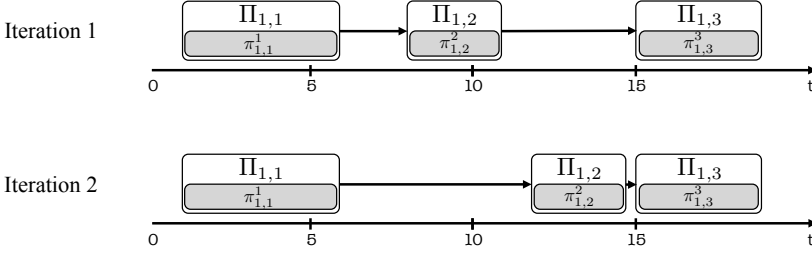
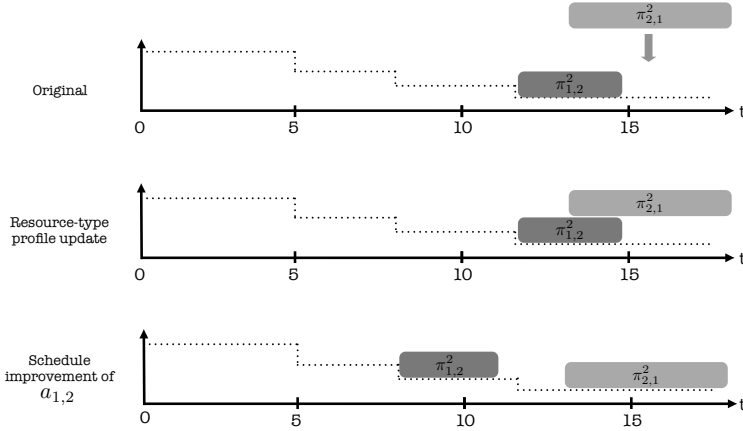
be further improved in iteration 2. Thus, the project schedule obtained in iteration 1 is currently the best schedule of PA_1 .

We note that updates of activity secure time windows can only be caused by the schedule changes of activities in the same project. Schedule changes of other projects will not influence the secure time windows of this project's activities. The reason is that precedence constraints only exist among activities of the same project. In the next subsection, we discuss how the ISIM can improve a project-agent utility when schedule changes of other projects occur.

5.2.2 ISIM by Resource-type-profile Update

Let us illustrate the ISIM in case there is an update of the resource-type profile. We describe the method with the help of the same example project depicted in Figure 5.1. This time we look at the timeline of the resource type R_2 (see Figure 5.7). We remind readers that R_2 is the resource type required by the activity $a_{1,2}$. In Figure 5.7, the areas under the dotted line are the resources leased by earlier scheduled activities.

Remember that in iteration 1 of the ISIM, activity $a_{1,2}$ (of which $\mu_{1,2} = \langle \{(2: 2)\}, 3 \rangle$) was (re)scheduled on R_2 at the slot $\pi_{1,2,2}^P = \langle [12, 15], (2: 2), a_{1,2} \rangle$. The resource price of $\pi_{1,2,2}^P$ is 300 (see Table 5.9). We assume that after $a_{1,2}$ has been (re-)scheduled in iteration 1, an activity $a_{2,1}$ ($\mu_{2,1} = \langle \{(2: 2)\}, 5 \rangle$) of a newly released project P_2 is then scheduled at the slot $\pi_{2,1,2}^P = \langle [13, 18], (2: 2), a_{2,1} \rangle$ (see Figure 5.7). The newly scheduled activity $a_{2,1}$ causes a profile update of the resource type R_2 . The resource-type-profile update

Figure 5.6: Improves project-agent schedule Π_1 using the ISIMFigure 5.7: Schedule improvement of $a_{1,2}$ when R_2 profile changes

changes the market in a way that the prices for the slot offers that overlap with the time interval $[13,18)$ will be raised.

When this resource-type-profile update occurs, RA_2 would prefer the already scheduled activities to reschedule to a lower resource-cost slot in order to minimise its resource levelling cost. To do so, RA_2 will multicast a list of new slot offers to the project agents of which the activities are scheduled at the slots overlapping with $[13,18)$. As a consequence, PA_1 is informed by a new list of slot offers for scheduling $a_{1,2}$ (see Table 5.13). Two issues are worth noticing in the new list of offers: (1) the prices of two offers (i.e., offer number 6 and offer number 7) have been raised, and (2) the current cheapest offer is the offer having a time interval $[8,11)$.

The changes in the offer prices imply that the slot $\pi_{1,2,2}^P = \langle [12, 15), (2 : 2), a_{1,2} \rangle$ that was leased by PA_1 is now worth 300 more ($600 - 300 = 300$). This gives the project agent PA_1 an incentive to *decommit* from the lease $\pi_{1,2,2}^P = \langle [12, 15), (2 : 2), a_{1,2} \rangle$, and *recommit* to a new lease $\pi_{1,2,2}'^P = \langle [8, 11), (2 : 2), a_{1,2} \rangle$. From the decommitment, PA_1 will receive a compensation of 600 from RA_2 . Adversely, from the recommitment, PA_1 pays 400 to RA_2 for the new lease $\pi_{1,2,2}'^P$. In all, PA_1 saves 200 by rescheduling $a_{1,2}$.

	Slot			Price
	A	I	R	
1	$a_{1,2}$	[6, 9)	(2: 2)	600
2	$a_{1,2}$	[7, 10)	(2: 2)	550
3	$a_{1,2}$	[8, 11)	(2: 2)	400
4	$a_{1,2}$	[9, 12)	(2: 2)	400
5	$a_{1,2}$	[10, 13)	(2: 2)	400
6	$a_{1,2}$	[11, 14)	(2: 2)	400
7	$a_{1,2}$	[12, 15)	(2: 2)	300

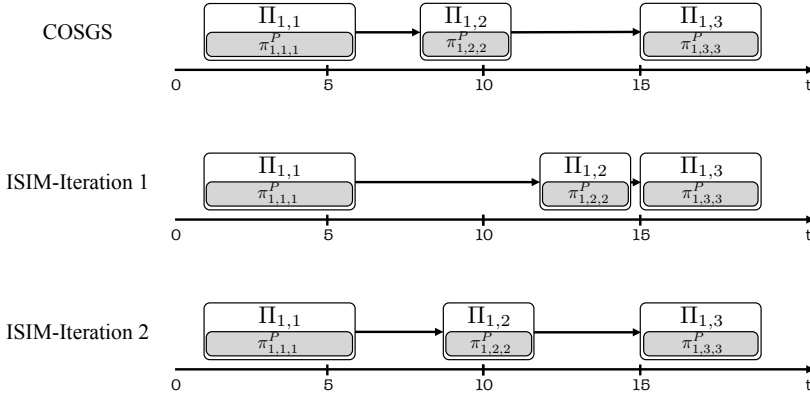
Table 5.11: Old offers sent by RA_2 \Rightarrow

	Slot			Price
	A	I	R	
1	$a_{1,2}$	[6, 9)	(2: 2)	600
2	$a_{1,2}$	[7, 10)	(2: 2)	550
3	$a_{1,2}$	[8, 11)	(2: 2)	400
4	$a_{1,2}$	[9, 12)	(2: 2)	400
5	$a_{1,2}$	[10, 13)	(2: 2)	400
6	$a_{1,2}$	[11, 14)	(2: 2)	600
7	$a_{1,2}$	[12, 15)	(2: 2)	600

Table 5.12: New offers sent by RA_2

Table 5.13: ISIM - Resource offers update

from $\langle [12, 15), (2: 2), a_{1,2} \rangle$ to $\langle [8, 11), (2: 2), a_{1,2} \rangle$, and the total cost of the schedule Π_1 in iteration 2 is 1400. The project-agent schedule obtained in iteration 2 is depicted in Figure 5.8.

Figure 5.8: Project-agent schedule Π_1 in iteration 2

Likewise, the rescheduling of activity $a_{1,2}$ causes again the resource-type-profile update of the resource type R_2 , this update creates new opportunities for earlier scheduled activities of other projects on resource type R_2 . We remind readers that the rescheduling of $a_{1,2}$ causes also the secure-time-window updates of $a_{1,1}$ and $a_{1,3}$. These updates create new opportunities to revise the project-agent schedule in the next iteration in order to increase PA_1 's utility. Therefore, the ISIM by resource-type-profile update leads to an indirect inter-project-agent interaction. Each project agent improves its schedule by the schedule changes of other project agents, and an improved global multi-project schedule emerges.

In the following section, we conduct experiments to evaluate the performance of the proposed OI-MAS scheduling approach.

5.3 Experiments

We remind the reader that the proposed OI-MAS scheduling approach consists of two components. (1) a clairvoyant online schedule generation scheme (COSGS) and (2) an iterative schedule-improvement method (ISIM). As we discussed in Section 5.1, when projects in a DRCPSP/u are scheduled in a COSGS, the presence of the first class of uncertainty — partial observability presented by variable project release times — cannot influence the obtained schedules. Therefore, the schedules made in the COSGS are **robust** under the variable-project-release uncertainty. This section focuses on investigating by experiments the **efficiency** of the schedules made by the proposed OI-MAS approach.

Since the COSGS is a single-pass scheduling scheme, the efficiencies of the schedules built by OI-MAS highly depend on the effectiveness of the schedule-improvement method (ISIM). In this section, we conduct experiments and empirically investigate two questions: (1) how effective is ISIM?, and (2) how efficient are the schedules constructed by OI-MAS? We first introduce the experimental setup (§5.3.1). Subsequently, we present and discuss the obtained results (§5.3.2).

5.3.1 Experimental Setup

The introduction to the experimental setup includes (a) a description of the problem instances, (b) performance criteria, and (c) the computational environment in terms of hardware and software.

A: Problem Instances

We employ two sets of problem instances: (1) a set of 80 benchmark problems extracted from the *Library for Multi-project Scheduling Problem* (MPSPLib³), and (2) a set of 10 simulated AGH scheduling problems. Below, we describe the properties of the two problem sets.

(1) problem instances from MPSPLib

MPSPLib is developed and maintained by Homberger (2007). The library comprises 140 multi-project problem instances. The number of projects (m) in each instance is 2, 5, 10, or 20. Each multi-project problem instance is made up of RCPSP instances from the *Library for Project Scheduling Problems* — PSPLib⁴, which is developed and maintained by Kolisch and Sprecher (1997). The multi-project problem instances differ in the composition of the chosen RCPSP instances as well as in the number of global resource types⁵ ($K = 1, 2, 3, 4$) and the release times of the projects.

We have chosen 80 problem instances from the MPSPLib for experimenting our MAS scheduling solution. The other 60 problem instances in MPSPLib are beyond our scope

³MPSPLib is accessible at <http://www.mpsplib.com/>. Last accessed on August 5, 2010.

⁴PSPLib is accessible at <http://129.187.106.231/psplib/>. Last accessed on July 15, 2009.

⁵Some problem instances in MPSPLib have so-called *local* resource types. Local resource types refer to the resource types of which the resource amount is dedicated to a single project. We decided to use problem instances with only global resource types to highlight the importance of the self-interested nature of resource agents.

of study because they consider problems with local resource types. The properties of the 80 chosen problem instances can be found in Appendix B: Properties of the chosen 80 MPSPLib Instances.

For brevity reasons, in the rest of this thesis, we will use the **alias** in Table B.1 to refer to the corresponding problem instance.

(2) simulated AGH scheduling problem instances

Besides experimenting with benchmark problems, we are also interested in investigating the performance of OI-MAS in simulated AGH scheduling problems. We simulate a set of AGH scheduling problems taking into account both the variety of aircraft turnaround processes and the difference in airport sizes.

First, we can distinguish several types of aircraft turnaround processes based on the differences in aircraft models (Boeing 747, Airbus 320, etc), aircraft usages (cargo or passenger aircraft) and docking locations (terminal gate or remote stand). Different turnaround processes may require different sets of ground handling operations. For instance, (i) the precedence relations among activities may be different, (2) same type of activities may require different types and amounts of resources, and (iii) activity processing durations may also be different in different turnaround processes. Nevertheless, ground handling activities performed in the same type of turnaround process share a large scale of similarities. Based on these observations, we simulate 10 types of aircraft turnaround processes by using 10 project instances ($J301.1 \rightarrow J301.10$) from PSPLib. One project instance consists of 30 activities, which resembles the number of ground handling operations required for an aircraft turnaround. The shared 4 resource types in an instance can be considered as 4 self-interested ground service providers. The optimal makespans for these 10 projects range from 30 to 60. We take the minute as the unit of time, these makespans resemble aircraft turnaround durations in actual situation.

Second, we consider a full range of airport sizes in terms of the number of aircraft movements. The busiest airport in the world, Atlanta International Airport in 2009, handled on average 1,300 aircraft a day (ACI, 2009). This means that, in every minute at peak hours, almost 2 aircraft land and another 2 take off. On the other hand, a small regional airport only handles a couple of aircraft a day.

According to the two types of varieties, we simulate a set of 10 instances of AGH scheduling problems (see Table 5.14).

B: Performance Criteria

The effectiveness of ISIM is evaluated by the improvement ratio of each project-agent schedule. The improvement ratio of a project-agent schedule measures how much the utility of a project agent is improved by applying ISIM.

The efficiency of schedules made by OI-MAS is evaluated by comparing the OI-MAS schedules to the schedules obtained by three other well known solution methods in the literature. Each of the three solution methods is chosen from a different solution category. They are (1) a centralised single-pass priority-rule-based heuristic approach — SASP by Lova and Tormos (2001), (2) a centralised forward-backward metaheuristic approach (Fwd/Bwd) by Lova et al. (2000), and (3) a game theoretical MAS approach (GT-MAS)

Instance	No. of operations per aircraft	No. of aircraft	No. of resource types	Project instances	Arrival period
IA2	30	2	4	$J301_1 \rightarrow J301_2$	30 mins
IA3	30	3	4	$J301_3 \rightarrow J301_5$	20 mins
IA6	30	6	4	$J301_5 \rightarrow J301_10$	10 mins
IA10	30	10	4	$J301_1 \rightarrow J301_10$	6 mins
IA20	30	20	4	$(J301_1 \rightarrow J301_10) \times 2$	3 mins
IA30	30	30	4	$(J301_1 \rightarrow J301_10) \times 3$	2 mins
IA40	30	40	4	$(J301_1 \rightarrow J301_10) \times 4$	2 per 3 mins
IA60	30	60	4	$(J301_1 \rightarrow J301_10) \times 6$	1 min
IA80	30	80	4	$(J301_1 \rightarrow J301_10) \times 8$	4 per 3 mins
IA120	30	120	4	$(J301_1 \rightarrow J301_10) \times 12$	2 per 1 min

Table 5.14: Simulated AGH Scheduling Problems

by Wauters et al. (2010). In general, each of the three solution methods produces the best schedules in their solution category.

The efficiency evaluation takes the form of two performance measures: (1) average project delay (APD in Equation 2.22) and (2) total squared resource utilisation (TSRU in Equation 2.14).

C: Computational Environment

To perform experiments, we implemented OI-MAS in a java-based agent programming platform — *Emerge* (a component of CHAP — Common Hybrid Agent Platform⁶). The experiments took place on a desktop computer with a 32bit 2.2GHz Intel Core 2 Duo processor and a 2G DDR2 RAM.

5.3.2 Results and Analysis

Below, we study the experimental results concerning both ISIM and OI-MAS as a whole. We subsequently investigate (a) the effectiveness of the schedule-improvement method ISIM, (b) the efficiency of the multi-project schedules constructed by the proposed OI-MAS approach, with comparison to the three other well-known solution methods, and (c) the flexibility of OI-MAS.

A: Effectiveness of ISIM

To evaluate the effectiveness of ISIM, we keep track of the schedules built by project agents in every iteration. The improvement ratio of a project-agent schedule is in terms of utility improvement of each project agent. We recall that the utility of a project agent is the negative value of its total cost — a combination of the project delay cost and the project’s resource cost (see Equation 5.5).

⁶CHAP can be found at <http://chap.sourceforge.net/>, last accessed on August 1, 2010.

$$f(S_i) = c_i^{dl} \cdot \max(s_{i,n_i+1} - \overline{dt_i}, 0) + \sum_{R_k \in \mathcal{R}} c_k^u \sum_{0 \leq t \leq \infty} u_k^2(S_i, t) \quad (5.5)$$

We assume that the project unit delay cost $c_i^{dl} = 100$ for all projects, and the resource unit utilisation cost $c_k^u = 1$ for all resource types. In reality, OI-MAS allows different (resource and project) agents to choose their own cost models.

Problem	Avg. Imprv. Ratio	Problem	Avg. Imprv. Ratio	Problem	Avg. Imprv. Ratio	Problem	Avg. Imprv. Ratio
I90/2/1	9.2%	I90/5/1	8.1%	I90/10/1	18.1%	I90/20/1	28.4%
I90/2/2	3.5%	I90/5/2	5.1%	I90/10/2	12.1%	I90/20/2	21.3%
I90/2/3	14.1%	I90/5/3	9.0%	I90/10/3	20.1%	I90/20/3	19.2%
I90/2/4	12.0%	I90/5/4	3.4%	I90/10/4	22.0%	I90/20/4	23.2%
I90/2/5	7.3%	I90/5/5	13.1%	I90/10/5	18.3%	I90/20/5	19.0%
I90/2/6	7.4%	I90/5/6	3.3%	I90/10/6	3.5%	I90/20/6	5.3%
I90/2/7	6.6%	I90/5/7	1.5%	I90/10/7	10.6%	I90/20/7	11.2%
I90/2/8	8.2%	I90/5/8	3.8%	I90/10/8	10.0%	I90/20/8	9.2%
I90/2/9	3.7%	I90/5/9	1.3%	I90/10/9	14.5%	I90/20/9	12.9%
I90/2/10	10.9%	I90/5/10	7.8%	I90/10/10	21.8%	I90/2/10	15.2%
I120/2/1	18.1%	I120/5/1	21.1%	I120/10/1	30.3%	I120/20/1	32.2%
I120/2/2	12.1%	I120/5/2	10.2%	I120/10/2	23.1%	I120/20/2	21.1%
I120/2/3	20.1%	I120/5/3	25.2%	I120/10/3	26.3%	I120/20/3	31.9%
I120/2/4	22.0%	I120/5/4	22.9%	I120/10/4	24.6%	I120/20/4	24.6%
I120/2/5	18.3%	I120/5/5	21.0%	I120/10/5	23.0%	I120/20/5	36.1%
I120/2/6	3.5%	I120/5/6	4.2%	I120/10/6	5.1%	I120/20/6	4.9%
I120/2/7	10.6%	I120/5/7	10.5%	I120/10/7	12.4%	I120/20/7	18.4%
I120/2/8	10.0%	I120/5/8	13.4%	I120/10/8	15.7%	I120/20/8	15.1%
I120/2/9	14.5%	I120/5/9	17.2%	I120/10/9	21.5%	I120/20/9	20.7%
I120/2/10	21.8%	I120/5/10	22.8%	I120/10/10	29.0%	I120/2/10	32.1%

Table 5.15: Average improvement ratios by ISIM on the 80 MPSPLib problem instances

Table 5.15 and Table 5.16 show the average improvement ratios by the ISIM of (i) the 80 MPSPLib problem instances and (ii) the 10 simulated AGH scheduling problems, respectively. From the average improvement ratios shown in the tables, we derive two observations as follows.

Observation 1 *All initial project schedules made in the COSGS are improved by the ISIM. On average, the improvement ratio is 15.1%.*

Observation 2 *The improvement ratios are dependent on the problem size. In general, more improvement can be obtained in large problems with more projects.*

Problem	Avg. Imprv. Ratio
IA2	2.2%
IA3	8.1%
IA6	8.7%
IA10	12.2%
IA20	14.6%
IA30	17.3%
IA40	18.2%
IA60	18.9%
IA80	21.0%
IA120	23.3%

Table 5.16: Average improvement ratios by ISIM on the simulated AGH instances

Problem	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
I90/10/1	37.6%	26.6%	19.8%	3.6%	28.1%	3.4%	9.1%	8.0%	26.0%	18.6%
I90/10/2	18.3%	4.3%	7.1%	34.1%	9.3%	8.9%	16.1%	11.8%	7.8%	3.8%
I90/10/3	56.8%	49.3%	12.7%	13.1%	9.84%	5.14%	26.7%	7.05%	19.7%	0.96%
I90/10/4	20.3%	18.4%	24.7%	4.4%	27.5%	12.9%	38.2%	37.3%	8.5%	27.1%
I90/10/5	30.3%	22.1%	25.0%	13.3%	13.9%	20.3%	20.6%	15.3%	7.2%	14.6%
I90/10/6	3.0%	2.6%	12.4%	1.7%	0.2%	6.5%	1.9%	1.6%	4.5%	1.1%
I90/10/7	3.3%	3.9%	5.6%	15.0%	6.2%	10.4%	38.5%	4.2%	13.9%	5.5%
I90/10/8	16.1%	35.5%	3.7%	2.3%	6.2%	5.4%	8.6%	10.7%	8.0%	2.9%
I90/10/9	2.4%	39.1%	27.8%	3.0%	11.7%	21.9%	13.5%	6.0%	3.7%	16.3%
I90/10/10	37.4%	1.4%	44.1%	13.2%	22.7%	43.9%	3.1%	14.4%	17.9%	21.0%

Table 5.17: Project-by-project improvement ratios by ISIM on 10 I90/10 instances

Furthermore, we have chosen 10 problem instances and looked into the improvement ratios project by project. Table 5.17 shows the project-by-project improvement ratios on the 10 I90/10 instances.

Observation 3 *The ISIM improvement ratio varies from project to project.*

In Table 5.17, we see that some project schedules can be improved significantly (e.g., 56.8% for project P_1 in problem instance I90/10/3). However, some others only make little improvement (e.g., 0.2% of project P_5 in problem instance I90/10/6). This is due to the difference in project release times, as well as the project structure itself. In general, the earlier the projects are released, the more iterations of improvement there are. This gives the earlier-released projects more chances for schedule improvement. However, it does not necessarily mean that more improvements are made within the earlier-released projects. We notice that the activity network and the resource requirements of a project play important roles of improvement ratios. In general, the more resource types/amounts an activity requires, the more chance there is that the activity schedule can be improved.

Based on the three observations listed above, we validate the effectiveness of the ISIM

in improving schedules made in the COSGS.

B: Efficiency of OI-MAS

In order to investigate the efficiency of OI-MAS, we first look at the number of iterations needed for ISIM in order to achieve a stable⁷ schedule.

Problem instances	Min	Max	Avg	Problem instances	Min	Max	Avg
I90/2	2	5	3	I120/2	4	9	5
I90/5	5	26	12	I120/5	7	28	14
I90/10	10	69	34	I120/10	13	99	46
I90/20	18	203	63	I120/20	20	289	69

Table 5.18: Minimal, maximal, and average numbers of iterations to achieve a stable schedule

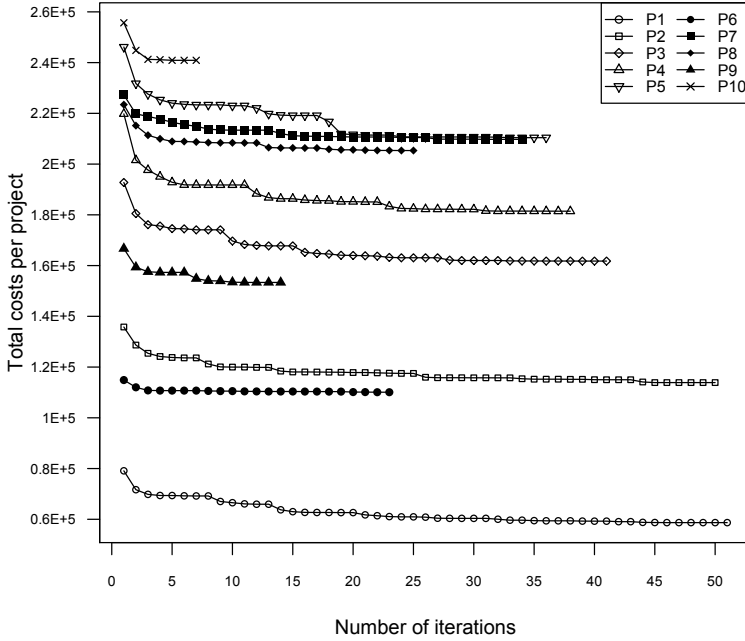


Figure 5.9: Schedule improvement by ISIM for the 10 projects in I90/10/1

Table 5.18 shows the minimal, maximal, and average numbers of iterations needed for ISIM to achieve a stable schedule. In additional, Figure 5.9 plots the evolution of the

⁷In this context, ‘stable’ means that no more improvement can be made by ISIM. Later in Chapter 6, we will introduce the notion of stability in a different context.

schedules improvement for all the 10 projects in problem instance I90/10/1. Based on Table 5.18 and Figure 5.9, we arrive at the following observation.

Observation 4 *After a small amount of iterations, agents can find stable schedules.*

When the scheduling time scale is critical, such as in the AGH scheduling environment, the number of iterations in order to reach a much improved schedule is crucial. From Table 5.18, we see that the maximum number of iterations to achieve a stable schedule is 289. In average, project agents require fewer than 70 iterations to reach a stable schedule. The small number of iterations demonstrates the efficiency of ISIM in making schedule improvement.

Below, we compare the efficiency of the obtained OI-MAS schedules to the state-of-the-art solution methods. We look at the schedules made for both the MPSPLib instances (see Table 5.19 and Table 5.20) and the simulated AGH instances⁸ (see Table 5.21). For the time-based objective (APD), we observe that two approaches (GT-MAS and Fwd/Bwd) outperform OI-MAS. Within these two approaches, GT-MAS performs the best in general. In all, GT-MAS, Fwd/Bwd, and OI-MAS outperforms the single-pass heuristic approach SASP. Nevertheless, with respect to the resource-based objective (resource levelling measure TSRU in our case), OI-MAS outperforms significantly all three other approaches and results in a lowest total squared resource utilisation (see Table 5.20). Based on observation No. 4 and the two tables above, we can conclude the efficiency of OI-MAS.

Problem instances	APD (percentage)			
	OI-MAS	GT-MAS	Fwd/Bwd	SASP
I90/2	135.5 (-9.64%)	103.15 (-31.2%)	110.3 (-26.4%)	149.95 (0)
I90/5	379.78 (-4.97%)	245.56 (-38.6%)	314.7 (-21.3%)	399.62 (0)
I90/10	227.92 (-16.4%)	169.37 (-37.9%)	201.89 (-25.9%)	272.6 (0)
I90/20	142.5 (-5.35%)	85.44 (-43.3%)	90.45 (-39.9%)	150.6 (0)
I120/2	51 (-7.27%)	35.2 (-36.1%)	37.9 (-31.1%)	55 (0)
I120/5	232 (-2.27%)	178.8 (-24.7%)	192.1 (-19.1%)	237.38 (0)
I120/10	139.1 (-3.90%)	114.33 (-21.0%)	125.1 (-13.6%)	144.74 (0)
I120/20	215.5 (-5.23%)	158.6 (-30.3%)	164.3 (-27.7%)	227.4 (0)

Table 5.19: Comparison of four methods on average project delay (APD)

C: Flexibility of OI-MAS

In OI-MAS, the decision on a lease commitment is dependent on the utility measurement of two agents. The case of two agents choosing a different utility measurement may result in two different global schedules. We conduct a series of 30 experiments using the

⁸Because the simulated AGH instances are not benchmark instances, there are no schedules made by other three approaches available for comparison. A special gratitude goes to Tony Wauters for generating the results by his GT-MAS approach.

Problem instances	TSRU (percentage)			
	OI-MAS	GT-MAS	Fwd/Bwd	SASP
I90/2	155492 (-19.5%)	198201 (+2.59%)	199343 (+3.18%)	193196 (0)
I90/5	440645 (-18.8%)	540851 (-0.38%)	542149 (-0.14%)	542908 (0)
I90/10	2884382 (-3.88%)	3044724 (+1.47%)	3015612 (+0.50%)	3000659 (0)
I90/20	13510595 (-32.5%)	21014051 (+4.99%)	20183423 (+0.84%)	20015696 (0)
I120/2	154613 (-26.4%)	216778 (+3.25%)	213464 (+1.67%)	209953 (0)
I120/5	698139 (-18.3%)	859159 (+0.50%)	858641 (+4.35%)	854925 (0)
I120/10	5692162 (-13.1%)	6744306 (+3.00%)	6698644 (+2.30%)	6548068 (0)
I120/20	18134135 (-14.3%)	21249026 (+0.39%)	21205962 (+0.19%)	21166573 (0)

Table 5.20: Comparison of four methods on total squared resource utilisation (TSRU)

Problem instances	APD		TSRU		No. of average iterations	
	OI-MAS	GT-MAS	OI-MAS	GT-MAS	OI-MAS	GT-MAS
IA2	26	6	13586	16908	3	4160
IA3	15	1.67	26234	38322	8	3532
IA6	18.2	4.57	64190	81887	16	6220
IA10	27.9	15.41	151087	180471	29	4688
IA20	56.4	44.1	382580	453284	55	3402
IA30	72.5	59.5	721869	844369	104	4780
IA40	91.3	77.0	1093058	1277400	137	5479
IA60	80.9	74.7	2472212	2953207	166	5338
IA80	86.6	76.2	4169612	5097988	229	3750
IA120	108.4	104.7	7851954	9318896	336	4778

Table 5.21: Comparison of APD and TSRU on simulated AGH instances

problem instance I90/10/1. In the experiments, we assume (1) project agents choose a same project unit delay cost ($c_i^{dl} = 100, \forall P_i \in \mathcal{P}$) in all the experiments, and (2) resource agents choose a different resource-unit-utilisation cost ($c_k^u = \{0, 1, \dots, 30\}, \forall R_k \in \mathcal{R}$) in each experiment. Figure 5.10 shows the evaluations of the 30 schedules in terms of APD and TSRU. We observe a clear trade-off between project delay and resource levelling. The higher the average project delay goes (cf. x-axis), the less total squared resource utilisation (cf. y-axis) will be. From these results, we see that agents employing the same scheduling approach (OI-MAS as a whole) can result in different schedules when they emphasise different values. This gives the self-interested parties, not only project agents but also resource agents, a natural way of making decisions themselves based on their own value systems. In all, the autonomous decision making brings flexibility in the scheduling processing of OI-MAS.

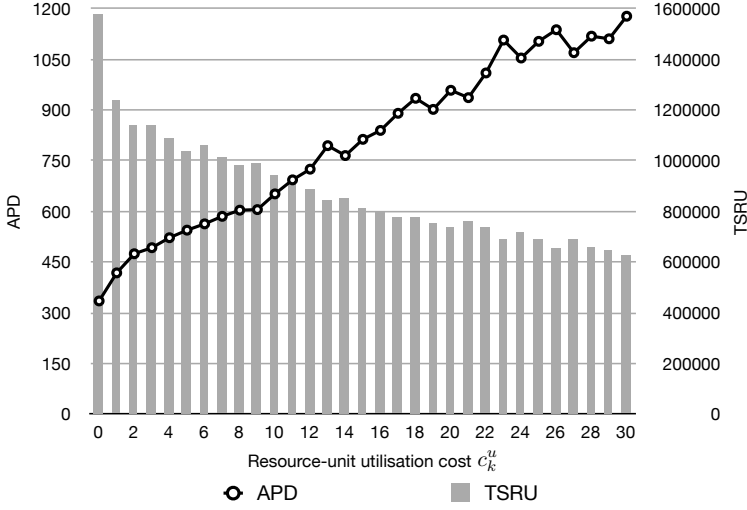


Figure 5.10: Trade-offs between project-agent objective and resource-agent objective

5.4 Answer to Research Question 2

In this chapter, we addressed our second research question. [RQ2: *How can agents make and coordinate their local decisions in order to achieve a globally efficient and robust schedule in a partially observable environment?*] Partial observability in project-scheduling context is presented by variable project release times.

We proposed OI-MAS, an online iterative multiagent scheduling approach. OI-MAS aims at constructing an efficient and robust multi-project schedule under variable-project-release-time uncertainty. Subsequently, validating both the efficiency and the robustness of the schedules constructed by OI-MAS can answer our research question 2.

OI-MAS deals with the efficiency and the robustness by its two components: (1) a clairvoyant online schedule generation scheme (COSGS), and (2) an iterative schedule-improvement method (ISIM).

Within the two components, the COSGS focuses on the robustness aspect of OI-MAS. In the COSGS, project agents adopt a clairvoyant online scheme in which the scheduling process of a project agent starts as soon as the project is actually released. Employing the COSGS effectively eliminates the possibility of schedule disruptions caused by the variable project release times. Evidently, the schedules obtained are robust with respect to the variable project release times.

Furthermore, the ISIM focuses on the efficiency aspect of OI-MAS. In the ISIM, project agents and resource agents revise iteratively their earlier determined schedules in order to increase their utilities. Based on the experiments and the analyses, we may conclude that the ISIM is effective, and the schedules constructed by OI-MAS are efficient.

In all, we can conclude that OI-MAS provides efficient and robust multi-project schedules in a partially observable project-scheduling environment.

Chapter 6

Stable Proactive Scheduling

In Chapter 5 we have seen that the clairvoyant online scheduling scheme employed by project agents is able to overcome the problem of having schedules disrupted by the first class of environmental uncertainty — variable project release times. However, the scheme given does not offer a solution for uncertainties raised on the processing times of activities (a.k.a., nondeterminism). In this chapter, we address problems characterised by the second class of uncertainty — variable activity processing times. Accordingly, we will answer research question 3, which reads as follows.

RQ3: *How can agents make and coordinate their local decisions in order to achieve a globally efficient and robust schedule in a nondeterministic environment?*

Given the uncertainty of variable activity processing times, the agents have to construct collaboratively schedules that, to a certain extent, are tolerant to minor incidents.

We recall that in Section 3.3, five classes of scheduling approaches for dealing with uncertainty are discussed. In this chapter, we extend our study on one of the five classes, i.e., the class of proactive-reactive scheduling approaches. We focus on developing an agent-based robust-proactive-scheduling approach.

The chapter is organised as follows. In Section 6.1, we introduce the concept of stability as a solution-robustness measure for project schedules. In Section 6.2, we present an agent-based solution model for constructing a stable (and yet efficient) multi-project schedule. In Section 6.3, we simulate DRCMPSPs/u with variable activity processing times and evaluate the performance of the proposed stable scheduling procedure. Based on the findings, Section 6.4 answers the research question 3.

6.1 Stability: Solution Robustness

In general terms, a **robust** decision is referred to as a decision that is immune to uncertainty and looks good to all constituents long after it is made. Robustness in the context of project scheduling under uncertainty has two major forms (cf. Sevaux and Sörensen,

2002): (1) robustness in the *objective-function space* (a.k.a. *quality robust*), and (2) robustness in the *solution space* (a.k.a. *solution robustness*). Below, we discuss briefly the two forms of robustness.

DEFINITION 6.1 Quality Robustness. *A schedule is called quality robust when it remains high quality in terms of the objective value when disruptions occur during project executions.*

Typical quality-robust schedules under uncertainty are achieved by building so-called *flexible* schedules that can be easily repaired, i.e., changed into new high-quality schedules — in terms of objective functions — whenever a disruption occurs.

DEFINITION 6.2 Solution Robustness. *A schedule is called solution robust when the activity start times in the schedule are insensitive to disruptions during project executions.*

Solution robustness is often known as **stability** (cf. Herroelen and Leus, 2004). It means that given the uncertainty during execution, one would like the realised schedule to resemble the expected schedule as much as possible. In this chapter we focus on solution robustness.

In the remainder of this section, we will first investigate how to measure the stability of a given project schedule¹ (see §6.1.1). In §6.1.2, we discuss the concept of stability in proactive-reactive project-scheduling procedures. Subsequently, §6.1.3 presents two classes of existing solution methods for stable proactive scheduling based on (1) organising resource flows between project activities, and (2) reserving extra slack time and/or resource capacity. Lastly, in §6.1.4, we discuss the (in)applicability of the approaches in an agent-based model for DRCMPSP/u.

6.1.1 Stability Measures

A commonly used stability measurement of a project schedule is proposed by Leus (2003) and Herroelen and Leus (2004) in a single-project environment. When a project is executed differently from it was scheduled, *instability cost* has to be paid. In practice, instability costs may include financial costs, inventory costs or various organisational costs. Leus (2003) and Herroelen and Leus (2004) measured the instability costs as the *weighted sum of the deviations* between the scheduled activity start times in the project schedule and the actually realised activity start times during project execution. They optimise the stability of a project schedule by minimising the instability cost. The expression to minimise is as follows.

$$\sum_{j=1}^n w_j (E(s_j^*) - s_j), \quad (6.1)$$

in which E is the expectation operator, s_j is the start time of activity a_j in the baseline schedule $S = \{s_1, s_2, \dots, s_n\}$, and s_j^* is a stochastic variable representing the **actually** achieved start time of activity a_j (after project execution). Consequently, the real project

¹Actually, it is the instability that is measured.

execution is a stochastic vector $S^* = \{s_1^*, s_2^*, \dots, s_n^*\}$. In Equation 6.1, w_j stands for the non-negative marginal cost per unit time overrun the scheduled start time of activity a_j .

This measure for instability cost in a single-project problem can be easily adapted to a measure in a multi-project problem. In Equation 6.7, we show the new measure.

$$\sum_{i=1}^m \sum_{j=1}^{n_i} w_{i,j} (E(s_{i,j}^*) - s_{i,j}), \quad (6.2)$$

where i is the index of a project, and j is the index of an activity in a project.

6.1.2 Stability in Proactive-reactive Scheduling Procedures

We recall that proactive-reactive scheduling is a two-stage scheduling procedure (see §3.3.1). In the first stage, proactive scheduling constructs a **baseline schedule** prior to the project start. Then in the second stage while executing the baseline schedule, reactive scheduling revises the schedule when it is invalidated by unexpected events. In this subsection, we discuss stability in the proactive scheduling procedure and in the reactive scheduling procedure, respectively. We refer to the two procedures as (a) stable proactive scheduling and (b) stable reactive scheduling.

A: Stable Proactive Scheduling

Stable proactive scheduling is also known as *fault-tolerant* scheduling. In a project-scheduling context, stable proactive scheduling aims at constructing a baseline project schedule (i.e., a project schedule proactively constructed prior to the project start) that incorporates anticipated project-execution variability. Having the property of stability for a baseline schedule in project-scheduling domain is more crucial than in any other scheduling domain. This is because a baseline project schedule is used to organise resources, negotiate contracts with sub-contractors (or service providers), etc. A stable baseline schedule is able to absorb some level of (un)expected disruptions during the project execution without any reactive scheduling (Davenport et al., 2001).

We should emphasise that no matter how much we try to protect the baseline schedule against possible disruptions during the building process of proactive scheduling, we can never totally eliminate the possibility of having a disruption that renders a stable baseline schedule infeasible. In order to restore the schedule feasibility, some un-executed activities have to be re-scheduled. Therefore, a proactive scheduling will always require a reactive component to deal with schedule disruptions that cannot be absorbed by the baseline schedule.

B: Stable Reactive Scheduling

The reactive scheduling action may be based on various underlying strategies. At one extreme, reactive scheduling may involve a full scheduling of the unexecuted activities when the baseline schedule is invalidated. Such an approach is referred to as a (*full*) *rescheduling* approach. A reactive scheduling procedure may, in principle, be capable of maintaining optimal solutions if an exact (re)scheduling algorithm is employed. However,

alongside the high computational effort required, the main disadvantage of this procedure is that the resulting schedule can differ completely from the original baseline schedule. In order to generate a reactive schedule that deviates from the original baseline schedule as little as possible, *stable reactive scheduling* is introduced.

The goal of stable reactive scheduling is to deliver quickly a new schedule that is ‘optimal’ (or near-optimal) in terms of minimum deviation from the baseline schedule. In practice, many constraints prevent a reactive scheduling from conducting the optimisation procedures. Below, we mention two constraints that exist in the AGH scheduling problem.

First, it is often undesirable to advance the baseline-scheduled starting time of an activity. In AGH domain, an aircraft, for instance, will not take off before its scheduled takeoff time. Even if it can be guaranteed that all passengers taking the flight are on board, the aircraft will normally not leave early in order to avoid upsetting the global air-traffic schedule. Other areas of real-life scheduling environments exist where activities are frequently not allowed to commence before their scheduled start (even though sometimes technically possible), e.g., course scheduling, sports timetabling, and railway scheduling.

Second, it is often required that resource allocation to activities remain constant, i.e., the same resource flow is maintained. Such a reactive scheduling policy is often preferred when transferring resources between activities are not achievable at short notice. In AGH domain, ground handling operations that require the same type of resources are carried out at different terminal gates. Transferring key staff or scarce equipment with high setup costs is often unwanted and sometimes unachievable.

The aforementioned two constraints in AGH domain render most reactive scheduling procedures inapplicable². In this thesis, we focus our research on constructing a stable proactive baseline schedule, and employ a *right-shift-rule* policy for reactive scheduling procedure (cf. Sadeh et al., 1993). When during execution, an activity encounters an incident that causes its processing time being longer than estimated, the right shift rule moves forward in time all the activities that are affected by this incident. The affected activities can be those (i) making use of the same resources or (ii) having precedence constraints with the activity.

6.1.3 Solution Models for Stable Proactive Scheduling

In this subsection, we present two solutions models for stable proactive scheduling. They are (a) the solution model based on organising resource flows and (b) the solution model based on allocating redundancies.

A: Organising Resource Flows

The first solution model for stable proactive scheduling is to organise resource flows between project activities. The way in which renewable resources are passed on between project activities can be represented by a *resource flow network*. Below, we define the network.

DEFINITION 6.3 Resource Flow Network. A resource flow network G_k of a resource type R_k is a directed graph $G_k = (V_k, E_k)$, which comprises a set of vertices V_k and a set

²A survey of reactive scheduling procedures can be found in van de Vonder et al. (2007).

of directed edges E_k .

$$V_k = \{v_s^k\} \cup \{v_{i,j}^k \mid r_{i,j}^k \in \mathbb{N}^+\} \cup \{v_t^k\}, \quad (6.3)$$

$$E_k = \{f_{v_{i,j}^k \rightarrow v_{i',j'}^k} \mid f_{v_{i,j}^k \rightarrow v_{i',j'}^k} \in \mathbb{N}^+\}. \quad (6.4)$$

In Equation 6.3, v_s^k and v_t^k are two vertices that represent the **source** and the **sink** of the resource flows, respectively. In addition, $v_{i,j}^k$ is a vertex representing the activity $a_{i,j}$ that requests resources of R_k for execution ($r_{i,j}^k \in \mathbb{N}^+$). In Equation 6.4, the flow quantity $f_{v_{i,j}^k \rightarrow v_{i',j'}^k} \in \mathbb{N}^+$ represents the amount of resource units of R_k passing on from activity $a_{i,j}$ (when it finishes) to activity $a_{i',j'}$ (when it starts).

A resource flow network must satisfy certain constraints in order to be called feasible. Below, we define a *feasible resource flow network*.

DEFINITION 6.4 A Feasible Resource Flow Network. *A resource flow network G_k is called feasible when it satisfies the following two constraints:*

$$\sum_{v \in V_k} f_{v_s^k \rightarrow v} = \sum_{v \in V_k} f_{v \rightarrow v_t^k} = \bar{c}_k, \quad (6.5)$$

$$\sum_{v_{i',j'}^k \in V_k} f_{v_{i',j'}^k \rightarrow v_{i,j}^k} = \sum_{v_{i'',j''}^k \in V_k} f_{v_{i,j}^k \rightarrow v_{i'',j''}^k} = r_{i,j}^k. \quad (6.6)$$

We remind the reader that in the thesis we consider only **renewable resources** (see Definition 2.4). It means that the amount of resources consumed by an activity at a time point t will be fully renewed at the next time point $t+1$. Subsequently, the two constraints in Definition 6.4 can be explained as follows. The first constraint in Equation 6.5 proclaims that the sum of all flows going *out* of the source vertex should be equal to the sum of all flows going *into* the sink vertex, and both the sums should be equal to the maximum resource capacity \bar{c}_k . The second constraint (see Equation 6.6) proclaims that for each intermediate vertex $v_{i,j}^k$, the sum of flows going into this vertex must be equal to the sum of flows going out of the vertex, which must be equal to the resource requirement of the activity $a_{i,j}$: $r_{i,j}^k$.

Artigues et al. (2003) introduce a simple method to generate a feasible resource flow network. The method extends p-SGS by iteratively rerouting flow quantities until a feasible overall flow is obtained. The method doesn't attempt to maximise schedule stability, thus will be used as a benchmark in the experiments in Section 6.3.

For a resource type R_k , it is often possible to make different resource allocation decisions for the same baseline schedule. Different allocation decisions will result in different resource flow networks. Below, we employ an example to illustrate the idea of resource-flow-based stable proactive scheduling.

On the left side of Figure 6.1, the AoN networks of two projects (P_1 and P_2) in a multi-project scheduling problem are shown. Each of two projects consists of four real activities ($A_1 = \{a_{1,1}, a_{1,2}, a_{1,3}, a_{1,4}\}$ and $A_2 = \{a_{2,1}, a_{2,2}, a_{2,3}, a_{2,4}\}$). The activity

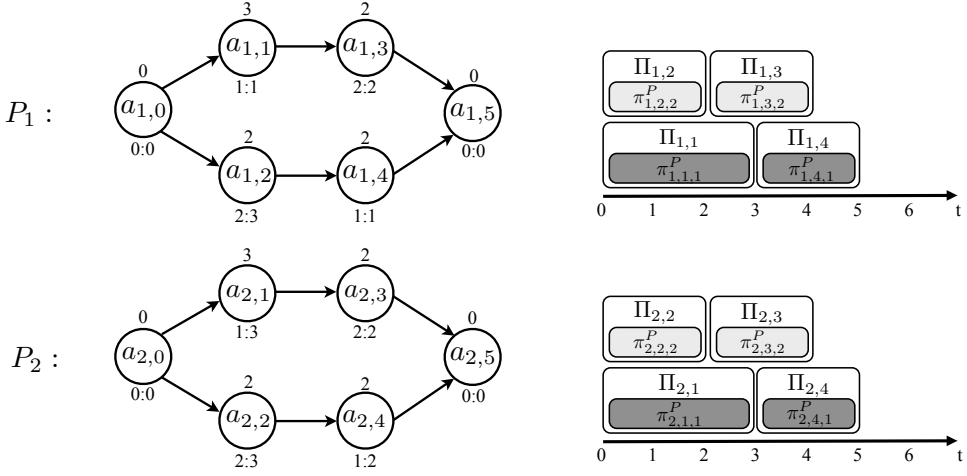


Figure 6.1: The AoN networks of two projects (left) and the project schedules (right)

modes are shown in the AoN networks. Two resource types R_1 and R_2 are required for processing the project activities. In this example, we focus on the resource-flow analysis of the resource type R_1 , which has a maximum capacity: $\bar{c}_1 = 4$.

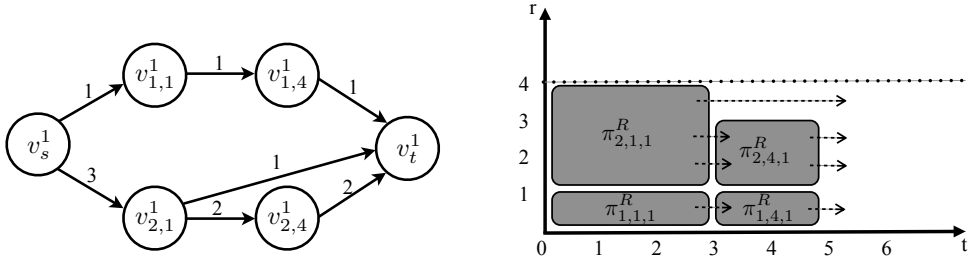


Figure 6.2: A resource flow network of R_1 and the corresponding resource profile

According to the schedules of the two projects shown on the right side of Figure 6.1, one possible resource flow of R_1 can be organised. The resource flow network is shown on the left side of Figure 6.2. Correspondingly, the right side of Figure 6.2 shows the scheduled resource profile of R_1 . The flows in the figures indicate that, at time point 0, one of the available resource units is transferred from the source vertex to $a_{1,1}$ ($f_{v_s^1 \rightarrow v_{1,1}^1} = 1$), and the rest three are transferred to $a_{2,1}$ ($f_{v_s^1 \rightarrow v_{2,1}^1} = 3$). Later at time point 3, when both $a_{1,1}$ and $a_{2,1}$ are finished, the resource unit used by $a_{1,1}$ is transferred to $a_{1,4}$ ($f_{v_{1,1}^1 \rightarrow v_{1,4}^1} = 1$); two of the three resource units used by $a_{2,1}$ are transferred to $a_{2,4}$ ($f_{v_{2,1}^1 \rightarrow v_{2,4}^1} = 2$); and the rest resource unit used by $a_{2,1}$ goes to the sink ($f_{v_{2,1}^1 \rightarrow v_t^1} = 1$). Finally, at time point 5, resource units used by $a_{1,4}$ and $a_{2,4}$ are all transferred to the sink vertex ($f_{v_{1,4}^1 \rightarrow v_t^1} = 1$ and $f_{v_{2,4}^1 \rightarrow v_t^1} = 2$).

Alternatively, for the same project schedules depicted on the right side of Figure 6.1, another resource flow can be organised (see Figure 6.3). The difference between the two resource flow networks lies at time point 3, where in the latter case the resource unit used by $a_{1,1}$ is transferred to $a_{2,4}$ ($f_{v_{1,1}^1 \rightarrow v_{2,4}^1} = 1$) instead of to $a_{1,4}$. In addition, the three resource units used by $a_{2,1}$ go different ways: one goes to $a_{2,4}$ ($f_{v_{2,1}^1 \rightarrow v_{2,4}^1} = 1$), one goes to $a_{1,4}$ ($f_{v_{2,1}^1 \rightarrow v_{1,4}^1} = 1$), and the last goes to the sink ($f_{v_{2,1}^1 \rightarrow v_t^1} = 1$).

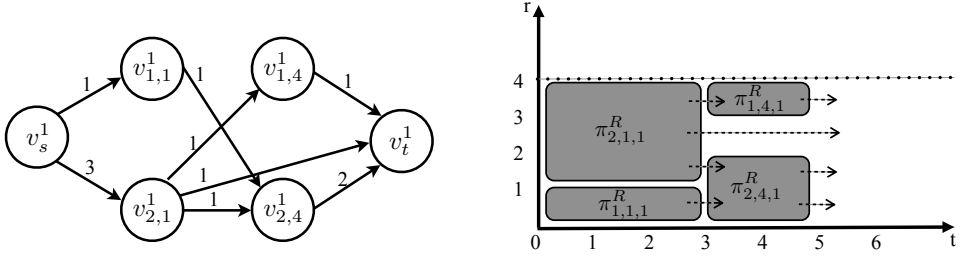


Figure 6.3: An alternative resource flow network of R_1 and the resource profile

The possibility of generating different resource flow networks for the same baseline schedule may have a serious impact on the stability of the baseline schedule. Let us assume that activity $a_{2,1}$ encounters a minor disruption during execution. The disruption causes a longer processing time of $a_{2,1}$ than it was estimated ($p_{2,1}^* > p_{2,1}$, where $p_{2,1}^*$ denotes the **actual** processing time of $a_{2,1}$). In case the resource flows of R_1 are organised according to Figure 6.2, only one activity's schedule (i.e., the one of $a_{2,4}$) has to be revised. The schedule of $a_{1,4}$ remains intact. However, if the resource flows of R_1 are organised according to Figure 6.3, both of the two activity schedules ($\Pi_{1,4}$ and $\Pi_{2,4}$) have to be revised, incurring higher instability costs.

Organising an optimal resource flow network for a single-project scheduling problem with a single disruption is proven to be NP-hard by Leus (2003). Undoubtedly, in practice, exact methods for large multiple-project problems with multiple disruptions are inapplicable. In order to deal with problem complexity, Deblaere et al. (2007) presented three heuristics for organising resource flows. They are (1) minimise the number of extra arcs (MinEA), (2) maximise the sum of pairwise floats (MaxPF), and (3) minimise the estimated disruption (MinED). These heuristics are based on surrogate *mixed-integer-programming* (MIP) formulation of the original strongly NP-hard problem.

B: Allocating Redundancies

Apart from organising resource flows, the second solution model for stable proactive scheduling is to allocating redundancies (protections). Redundancy often takes form of slack in time or (extra-)capacity in resource. By reserving extra time and/or resources, the baseline schedule is able to absorb locally some level of expected (or unexpected) disruptions during project execution without cascading the disruption to a larger scale (cf. Davenport et al., 2001).

However, for AGH scheduling problems in which we are interested, pure resource

redundancy is unrealistic (cf. Davenport and Beck, 2000). We give two main reasons. First, the resource redundancy is achieved by allocating multiple identical sets of resources for an activity (cf. Ghosh, 1996). The cost of providing redundant resources is so high that almost none of the service providers would double its resource inventory. Second, allocating resource redundancies is often not practical in problems with decentralised decision-making processes and large geometric span. In such problem settings, in case no disruption occurs, the extra reserved resources cannot be easily re-allocated to other activities at a short notice.

In contrast, time redundancy is relevant. Allocating time redundancies is achieved by adding *slack time windows* before (or after) the schedules of individual activities. Below, we employ an example to illustrate the idea of slack-time-based stable proactive scheduling.

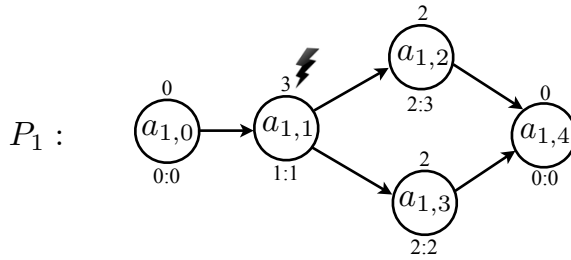


Figure 6.4: The AoN network of a project P_1 with a disrupted activity $a_{1,1}$

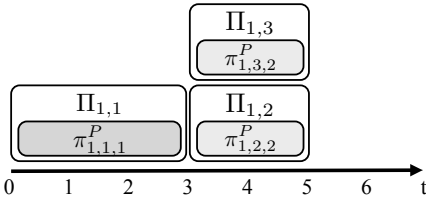


Figure 6.5: Schedule Π_1 without slack time

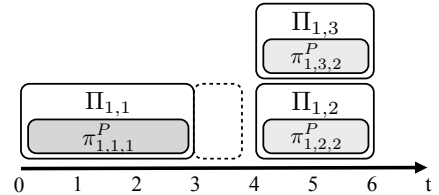


Figure 6.6: Schedule Π'_1 with a slack time

Figure 6.4 depicts the AoN network of a project P_1 consisting of three real activities ($A_1 = \{a_{1,1}, a_{1,2}, a_{1,3}\}$). Figure 6.5 shows an optimal — in terms of minimum project makespan — schedule Π_1 of P_1 . An alternative schedule Π'_1 with a one-time-unit slack time window inserted after $\pi_{1,1,1}^P$ is shown in Figure 6.6. Let us assume that a minor disruption occurs while processing activity $a_{1,1}$. The disruption causes $a_{i,j}$ to be processed one time unit longer than estimated ($p_{1,1}^* = p_{1,1} + 1$). When the disruption occurs, the earlier considered optimal schedule Π_1 has to be revised. The revision requires that both of the two successors of $a_{1,1}$ be rescheduled, causing high instability costs.

In contrast, although the schedule Π'_1 is suboptimal compared to Π_1 in terms of minimum project makespan, the added one-time-unit slack time window $[3, 4]$ helps to absorb the disruption, and keeps the schedules of $a_{1,2}$, and $a_{1,3}$ intact. Thus, Π'_1 is more

stable than Π_1 in the sense that it is less sensitive to small fluctuations in the activity duration of $a_{1,1}$. With a slack time protected schedule, disruptions will likely have local impact.

In slack-time-based approaches, the length of the slack time can either be analytically determined by a linear programming solver in case the disruption model is known or more practically determined by heuristic procedures.

For analytical approaches, Lambrechts et al. (2010) provided an overview of approaches for determining the expected duration increase an activity experiences due to resource breakdowns. We argue that in an agent-based model for DRCMPSP/u, project agents analytically determining the optimal slack time windows for its activities is an impossible task. In order to fulfil this task agents must not only know the pattern and frequency with which incidents occur, but also the resource requirement of other project agents. To make things worse, other project agents may also try to insert an optimal slack time after each project.

For heuristic procedures, Leus (2003) proposed a two-stage procedure, where at the first stage, a (sub)optimal deterministic schedule is built; and at second stage, slack times are determined by a heuristic called *activity-dependent float factor* and inserted **in front of** activities. The problem of applying this two-stage solution procedure is the need of an almost complete re-scheduling at the second stage. When the problem is modelled in an agent-based model, re-scheduling an activity means de-committing a set of earlier reserved slots, and recommitting a new set of slots. In case this re-scheduling is not favourable to the resource managers, they may forbid this move in extreme cases or impose some penalty cost (often known as decommitment penalty) for this change. Therefore, a practical agent-based solution model for project managers is to pre-estimate the slack time and construct a schedule with the estimated slacks.

We notice that in a project scheduling problem where no deadline constraint is introduced, slack time windows can be large enough if one is only interested in keeping the baseline schedule stable instead of being interested in efficiency.

6.1.4 Towards an Agent-based Stable Scheduling

Various approaches of the two solution models have been proposed. In this subsection, we discuss how to adapt the solution models in an agent-based scheduling model.

In an agent-based model for DRCMPSP/u, the information about intra-project inter-activity precedence relations are not known to resource agents. The information asymmetry prevents resource agents from employing a slack-time-based solution to achieve a stable proactive baseline schedule. Likewise, the resource flow networks are kept “private” to the resource agents themselves. The project agents, on the other hand, will not be able to pursue a stable baseline schedule by organising resource flows. In this subsection, we discuss the (in)applicability of existing slack-time-based approaches and resource-flow-based approaches in an agent-based model for DRCMPSP/u .

Despite the attempts of reducing problem complexity by the three heuristics, solving the surrogate MIP problems is still computationally expensive (see Deblaere et al., 2007). For large multiple-project problems with multiple disruptions, constructive procedures are needed. Deblaere et al. (2007) proposed a single-pass constructive procedure, namely

myopic activity-based optimisation (MABO). Unlike most resource-allocation procedures, MABO works activity-based rather than resource-based, which means that inter-activity precedence relations are used to construct the resource flow. The activity-based procedure of MABO makes it inapplicable in an agent-based model, where inter-activity precedence relations are not known to resource agents.

In this thesis, we investigate approaches for different types of agents. Agent using different approaches to improve the schedule stability should be possessing the information that are available to them. In the following section, we present our agent-based stable proactive scheduling approaches.

6.2 Agent-based Stable Proactive Scheduling

In this section, we propose an agent-based stable proactive scheduling procedure, in which two classes of agents employ different methods to construct stable proactive schedules. First, resource agents organise their resource flows to make the schedules stable (see §6.2.1). Second, project agents insert time redundancies (in the form of slack time windows) after its activity schedules to protect the schedule stability(see §6.2.2).

6.2.1 Constructive Heuristic Procedures by Resource Agents

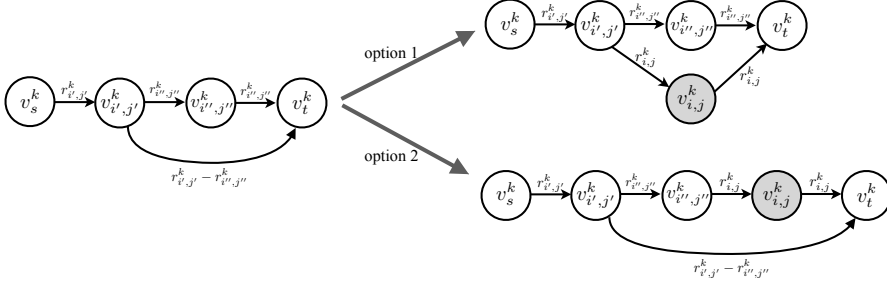
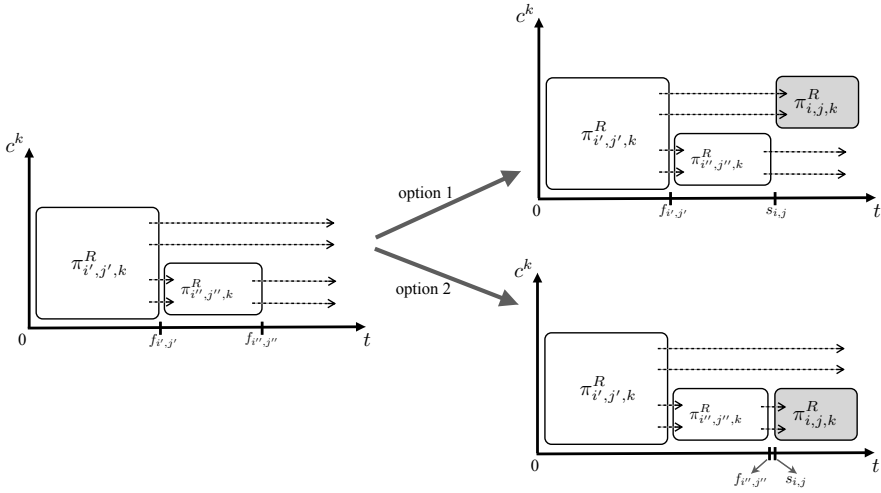
We present three constructive heuristic approaches for resource agents allocating resources to activities. They are (a) earliest finished predecessor first (EFPF), (b) richest predecessor first (RPF), and (c) cohabited predecessor first (CPF). Below, we describe the three heuristics individually.

A: Earliest Finished Predecessor First

Let us assume that in a resource flow network of resource type R_k , the amount of resources $r_{i,j}^k$ used by an activity $a_{i,j}$ can be obtained in two different ways: (i) it can be obtained totally from activity $a_{i',j'}$ ($f_{v_{i',j'} \rightarrow v_{i,j}}^k = r_{i,j}^k$) or (ii) it can be obtained totally from activity $a_{i'',j''}$ ($f_{v_{i'',j''} \rightarrow v_{i,j}}^k = r_{i,j}^k$). The two different ways of organising resource flows result in two different resource flow networks (see Figure 6.7). The resource profiles of the two options are shown in Figure 6.8.

Let the expected finishing times of the two scheduled activities ($a_{i',j'}$ and $a_{i'',j''}$) be $f_{i',j'}$ and $f_{i'',j''}$ ($f_{i',j'} < f_{i'',j''}$), respectively. In case the activity $a_{i',j'}$ undergoes a minor incident during its execution, the actual processing time $p_{i',j'}^*$ will be larger than it was expected ($p_{i',j'} < p_{i',j'}^*$). Let us assume that the actual finishing time $f_{i',j'}^*$ ($f_{i',j'}^* = s_{i',j'} + p_{i',j'}^*$) of $a_{i',j'}$ does not exceed the starting time $s_{i,j}$ of $a_{i,j}$ ($f_{i',j'}^* < s_{i,j}$). If the resource flows are organised according to the first option, the disruption of $a_{i',j'}$ will not influence the schedule of $a_{i,j}$. Instead, if the resource flows are organised according to the second option, both the schedules of $a_{i'',j''}$ and $a_{i,j}$ will be invalidated, incurring higher instability cost. Therefore, the resource flows in the first option is more stable than the one in the second.

Although resource agent RA_k cannot deliberately insert slack time window in between two activities, the difference between the scheduled finish time of $a_{i',j'}$ and the scheduled

Figure 6.7: Two options of obtaining resources for $a_{i,j}$ in a resource flow networkFigure 6.8: Two options of obtaining resources for $a_{i,j}$ in a resource-profile diagram

start time of $a_{i,j}$ (i.e., $f_{i',j'} - s_{i,j}$) in the first resource-flow option acts like a slack time window that protects the stability of the schedule. Thus, the larger the difference is, the more stable the resource-agent schedule will be. Based on this reasoning, we can propose a first constructed heuristic. We refer it to as *earliest finished predecessor first* (EFPF). In EFPF, a resource agent while organising the flow-in resources for an activity, will first choose the resources from the activity that has the earliest finishing time.

B: Richest Predecessor First

In a resource flow network, the edges show the interdependencies between activities in terms of resource flows. When an activity undergoes an incident, the number of edges going **out** of the activity's vertex indicates the possible number of activities of which the schedules might be invalidated. Likewise, the number of edges going **into** an activity's vertex indicates the likelihood of the activity being affected by earlier disruptions. Thus,

it is clear that in general the less the number of edges is in a resource flow network, the more stable the schedule will be.

Based on this reasoning, we can propose a second constructive heuristic. We refer it to as *richest predecessor first* (RPF). In RPF, a resource agent while organising the flow-in resources for an activity, will first choose the activity that has the most non-transferred resources. Below, we illustrate the RPF heuristic by an example.

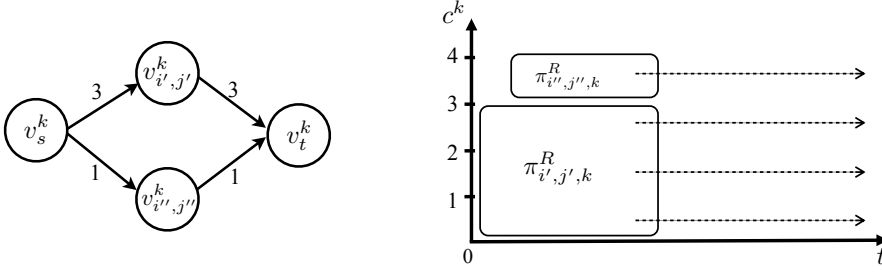


Figure 6.9: The resource flow network and the resource profile diagram of RA_k

Let us assume that two activities ($a_{i',j'}$ and $a_{i'',j''}$) have been scheduled on resource agent RA_k . The resource flow network and the resource profile diagram of RA_k are shown in Figure 6.9. An activity $a_{i,j}$ is scheduled on RA_k and there are two possibilities of allocating resources to $a_{i,j}$ (see Figure 6.10 and Figure 6.11).

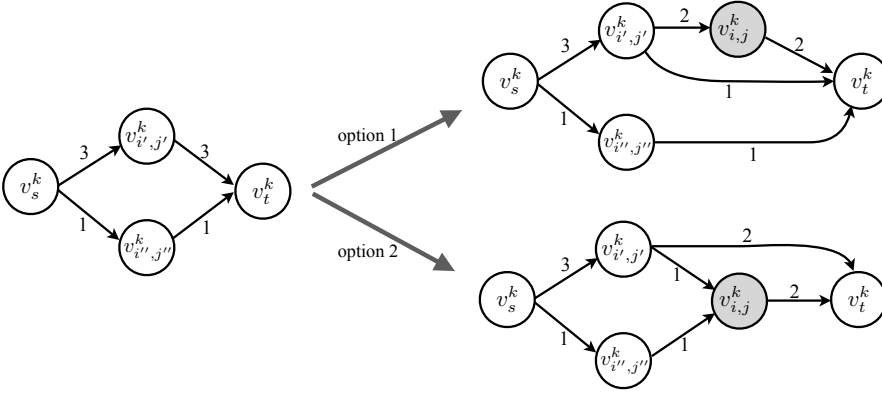


Figure 6.10: Two options of allocating resources for $a_{i,j}$ in resource flow networks

It is clear that the first option of organising resource flows is more stable than the second. Since the schedule of $a_{1,2}$ in the second option is vulnerable to disruptions to both activities $a_{i',j'}$ and $a_{i'',j''}$, while in the first option, it is only vulnerable to the schedule disruption of $a_{i',j'}$.

RPF tries to reduce the number of edges in a resource flow network. In consequence, interdependencies between activities imposed by resource-allocation decisions are reduced.

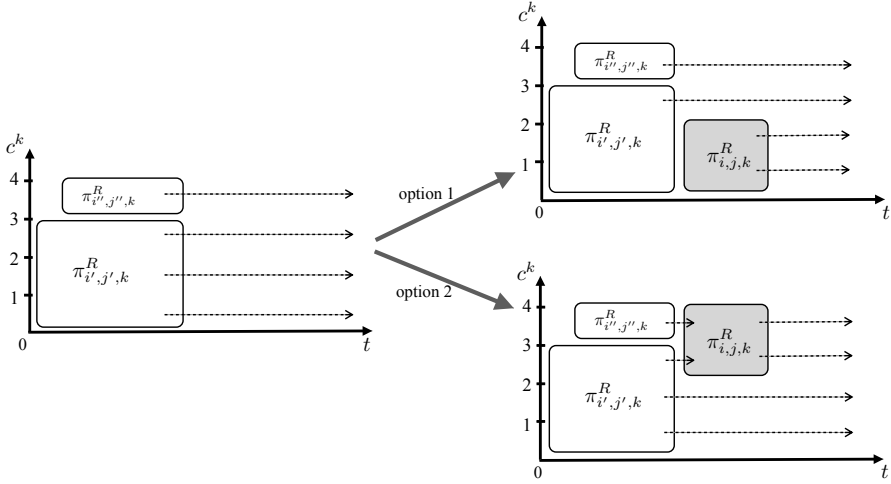


Figure 6.11: Two options of allocating resources for $a_{i,j}$ in resource-profile diagrams

C: Cohabited Predecessor First

As we discussed in RPF, the edges in a resource flow network create resource-related interdependencies among activities. Apart from resource-related interdependencies, we remind the reader that there is another type of activity interdependencies — the precedence relations (see §2.1.2). Precedence relations are inherent interdependencies once a problem is known. They are represented by arcs in AoN networks. Since precedence relations are inherent, they are *unavoidable* interdependencies.

Based on this reasoning, we can propose a second constructive heuristic. We refer it to as *cohabited predecessor first* (CPF). Two activities are called cohabited when they belong to the same project. In CPF, a resource agent while allocating resources for an activity, will first choose the resources that are released from activities belonging to the same project. Below, we illustrate the RPF heuristic by an example.

Let us assume that two activities ($a_{i,j'}$ and $a_{i',j'}$) have been scheduled on resource agent RA_k , both activities require two units of resources. A newly scheduled activity $a_{i,j}$, of which the resource requirement is also two units, belongs to the same project as the activity $a_{i,j'}$. Similar to earlier examples, there are also two possibilities of allocating resources to $a_{i,j}$ (see Figure 6.12 and Figure 6.13).

The two options seem equally stable judging from the number of edges in the resource flow networks. However, in case $a_{i,j}$ and $a_{i,j'}$ are precedence related ($a_{i,j'} \prec a_{i,j}$), a disruption occurring at the execution of $a_{i,j'}$ will also invalidate the schedule of $a_{i,j}$. Therefore, the first option is more stable than the second. Since the resource flow in the second option is vulnerable to both disruptions occurring in $a_{i,j'}$ and $a_{i',j'}$.

We note that although the resource agent RA_k cannot guarantee the precedence relation between the two activities³ ($a_{i,j}$ and $a_{i,j'}$), choosing a cohabited activity as a resource

³Intra-project precedence relations are only known to the corresponding project agent.

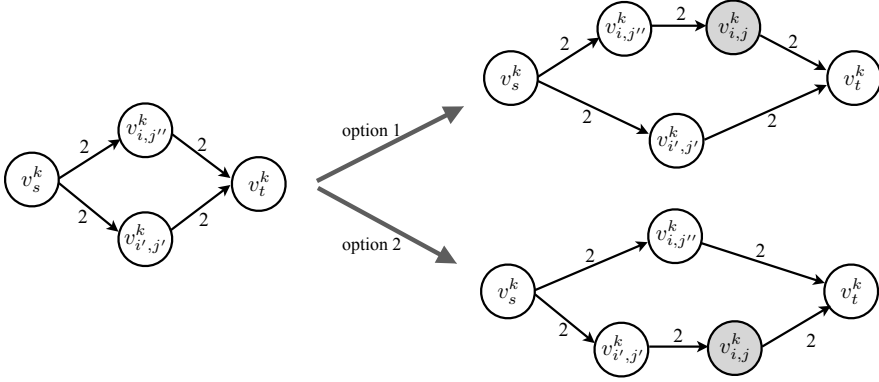


Figure 6.12: Two options of allocating resources for $a_{i,j}$ in resource flow networks

source increases the chance of reducing the number of resource-related interdependencies. CPF tries to attach resource-related interdependencies on the existing precedence-related interdependencies, therefore making the consequence of a disruption as local as possible.

6.2.2 Coevolving Slack Time Windows by Project Agents

For a project agent, stable proactive scheduling means to minimising the instability costs of its project schedule. According to the stability measure shown in Equation 6.1, the objective function for stable scheduling for a project agent PA_i can be represented by the following function.

$$\arg \min_{\vec{t}_i^{slk}} f(\vec{t}_i^{slk}) = \sum_{j=1}^n w_{i,j} (E(s_{i,j}^*) - s_{i,j}), \quad (6.7)$$

where $\vec{t}_i^{slk} = \langle t_{i,1}^{slk}, \dots, t_{i,n_i}^{slk} \rangle$ is a vector of n_i (the number of real activities of P_i) lengths of slack time windows. A slack time window with a length $t_{i,j}^{slk}$ is inserted after the schedule $\Pi_{i,j}$ of $a_{i,j}$ during the scheduling process.

As discussed in §6.1.4, analytically determining the optimal slack time windows by a project agent for all its activities is impossible. First, activities are not all equally likely to be disturbed and will not have the same expected disturbance length. Second, some activity start times may need to be better protected than others, for instance because of critical position of the activity in an AoN network. Third, when slack is spread out evenly, propagation of a disturbance throughout the network is not taken into account: an activity can not only be disturbed by delays in its immediate predecessors, but also because of disruptions of its transitive predecessors that could not be completely absorbed before reaching the activity.

Hence, multiple project agents determining optimal slack times of their own activities becomes a strategic decision game with no prior information about the payoff matrix. By playing the game the project agents can gain some information about their own rewards

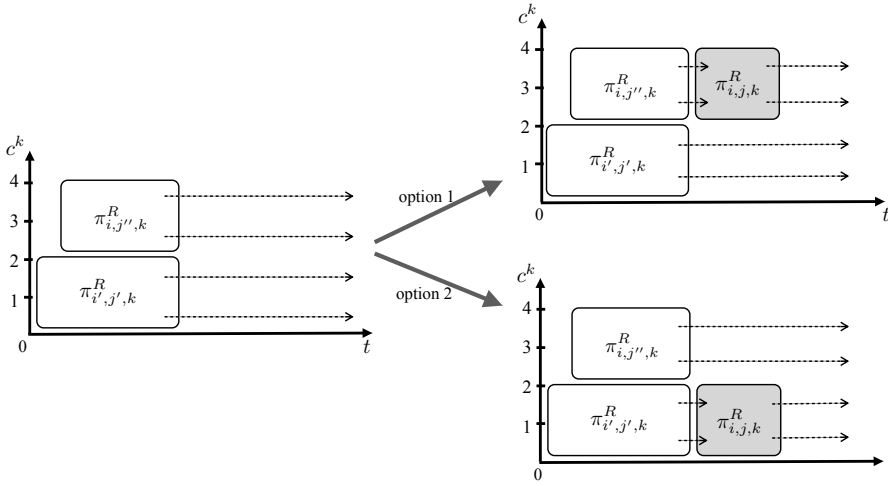


Figure 6.13: Two options of allocating resources for $a_{i,j}$ in resource-profile diagrams

although these rewards still depend on how other project agents play the game. Many multiagent learning techniques require that the payoff matrix is known by the agents. Techniques that can still be used when the payoff matrix is not known in advance are for instance *evolutionary game theory* (EGT, see Weibull (1997), Nash Q-learning (see Hu and Wellman, 2003), and *coevolutionary algorithms* (Co-EAs, see Paredis (2000)). In all cases convergence to a stable state is an important issue. We have chosen the use Co-EAs for the following reasons. First, we are interested in a proof of concept viz. can appropriate slack times be learnt. Second, we expect Co-EAs to be less sensitive for requirements that must be met to guarantee convergence. In the remainder of the subsection, we describe the Co-EAs for agents learning appropriate slack times.

In the Co-EAs, each project agent is equipped with an *evolution strategy* (ES) for learning a proper vector of slack-time-window lengths. All project agents together in an AGH scheduling ecosystem are coevolving their individual strategies.

ES is a subclass of EAs that use selection, recombination, and mutation to iteratively reproduce better-fit offsprings. Instead of encoding linear **binary** genotypes as in GAs, ES encodes problem-specific linear **real-valued** genotypes and typically uses self-adaptive mutation rates. Below, we propose a self-adaptive (1,1)-ES used by a project agent to learn a proper vector of slack time windows⁴.

Algorithm 6.1 describes the self-adaptive (1,1)-ES used by project agent PA_i . In the remainder of the section, we discuss the elements of the ES. The elements include (a) genotype and phenotype, (b) mutation, (c) fitness evaluation, and (d) initialisation and termination.

⁴The canonical notation of an ES is denoted by $(\mu/\rho, \lambda)$ -ES or $(\mu/\rho + \lambda)$ -ES, where μ denotes the total number of individuals in the population, ρ denotes the number of selected parents for reproduction, and λ denotes the number of offsprings. The *comma* sign means that the new generation is selected only from the best fitting offsprings. The *plus* sign means that the new generation is selected from the best fitting individuals from both the parents and the offsprings.

Algorithm 6.1 Self-adaptive (1,1)-Evolution Strategy used by PA_i

```

1:  $t := 0$ ;
2: Create an initial individual  $\langle \alpha_{i,1}^t, \dots, \alpha_{i,n_i}^t, \sigma_{i,1}^t, \dots, \sigma_{i,n_i}^t \rangle$ ;
3: for all  $j \in \{1, \dots, n_i\}$  do
4:    $t_{i,j}^{slk,t} := \lceil \alpha_{i,j}^t \cdot p_{i,j} \rceil$ ;
5: end for
6: repeat
7:   for all  $j \in \{1, \dots, n_i\}$  do
8:      $\sigma_{i,j}^{t+1} := \sigma_{i,j}^t \cdot e^{\tau_i \cdot \mathcal{N}_{i,j}(0,1)}$ ;
9:      $\alpha_{i,j}^{t+1} := \alpha_{i,j}^t + \sigma_{i,j}^{t+1} \cdot \mathcal{N}_{i,j}(0,1)$ ;
10:     $t_{i,j}^{slk,t+1} = \lceil \alpha_{i,j}^{t+1} \cdot p_{i,j} \rceil$ ;
11:   end for
12:   if  $f(\vec{t}_{i,j}^{slk,t}) \leq f(\vec{t}_{i,j}^{slk,t+1})$  then
13:     for all  $j \in \{1, \dots, n_i\}$  do
14:        $\sigma_{i,j}^t := \sigma_{i,j}^{t+1}$ ;
15:        $\alpha_{i,j}^t := \alpha_{i,j}^{t+1}$ ;
16:     end for
17:   end if
18:    $t := t + 1$ ;
19: until termination condition

```

A: Genotype and Phenotype

Two vectors of variables are used to encode an individual (genotype) for the project agent PA_i : (1) a vector of n_i *object variables*: $\vec{\alpha}_i = \langle \alpha_{i,1}, \dots, \alpha_{i,n_i} \rangle$ and (2) a vector of n_i *strategy variables*: $\vec{\sigma}_i = \langle \sigma_{i,1}, \dots, \sigma_{i,n_i} \rangle$. Altogether, an individual for the project agent PA_i is encoded as a vector of $2 \times n_i$ real numbers.

$$\underbrace{\langle \alpha_{i,1}, \dots, \alpha_{i,n_i} \rangle}_{\vec{\alpha}} \underbrace{\langle \sigma_{i,1}, \dots, \sigma_{i,n_i} \rangle}_{\vec{\sigma}} \quad (6.8)$$

An object variable $\alpha_{i,j} \in \mathbb{R}_{\geq 0}$ is a problem-related scaling factor that determines the length of the slack time window $t_{i,j}^{slk} \in \mathbb{N}$ (phenotype) to be inserted after the schedule of activity $a_{i,j}$. The length of the slack time window is computed as follows.

$$t_{i,j}^{slk} = \lceil \alpha_{i,j} \cdot p_{i,j} \rceil, \quad (6.9)$$

where $p_{i,j}$ is the estimated processing duration of $a_{i,j}$, and $\lceil x \rceil$ is a ceiling function returning the smallest integer no less than x .

A strategy variable $\sigma_{i,j} \in \mathbb{R}$ is a mutation parameter that represents the mutation step size of the objective variable $\alpha_{i,j}$. In ESs, the step sizes are also encoded in the genotypes and they themselves undergo variation and selection. This encoding gives ESs an important feature namely self-adaptation. Later in this section, we will describe how mutation works in the chosen ES.

B: Mutation

Mutations in (1,1)-ES are realised by adding some $\Delta\alpha_{i,j}$ to each $\alpha_{i,j}$. $\Delta\alpha_{i,j}$ values are randomly drawn using a given normal distribution $\mathcal{N}(\mu, \sigma)$, where μ is the mean and σ is the standard deviation (mutation step size). In practice, the mean μ is always set to zero and the vector $\vec{\alpha}_i$ is mutated by replacing $\alpha_{i,j}$ by

$$\begin{aligned}\alpha'_{i,j} &= \alpha_{i,j} + \mathcal{N}(0, \sigma) \\ &= \alpha_{i,j} + \sigma \cdot \mathcal{N}(0, 1)\end{aligned}$$

In a self-adaptive ES, the mutation step size σ is not set by the user, rather, it is also evolving. As specified in the individual encoding, we have chosen for each objective variable $\alpha_{i,j}$ an evolving mutation parameter $\sigma_{i,j}$. The mutation mechanism is thus specified by the following formulas.

$$\sigma'_{i,j} = \sigma_{i,j} \cdot e^{\tau_i \cdot \mathcal{N}_{i,j}(0,1)} \quad (6.10)$$

$$\alpha'_{i,j} = \alpha_{i,j} + \sigma'_{i,j} \cdot \mathcal{N}_{i,j}(0,1) \quad (6.11)$$

In Equation 6.10, the proportionality constant τ_i is an external parameter to be set by the user. It can be interpreted as a kind of *learning rate*. It is usually inversely proportional to the square root of the problem size:

$$\tau_i \propto 1/\sqrt{n_i}.$$

We refer the reader to the work by Bäck (1996) for more detailed discussion on the choice of τ_i .

C: Fitness Evaluation

In order to determine the fitness value of each individual, a project agent will calculate the lengths of the slack time windows based on Equation 6.9. The slack time windows are incorporated in the negotiation process with resource agents. By running simulations, the stability of each project is obtained (see Equation 6.1).

We remind readers that project-deadline constraint is not considered in our AGH scheduling problems (see Definition 2.11). In this case, a project agent who is purely interested in the schedule stability will allocate maximum slack time window for each of its activities. Allocating maximum slack time windows without deadline constraint will create a 100% stable project-agent schedule, meaning the instability cost equals to 0. However, the efficiency (measured in terms of project delay and resource utilisation cost) of the schedule will be immensely decreased. Therefore, a project agent should consider a trade-off between schedule efficiency and schedule stability. This leads us to the following fitness function of a project agent PA_i (see Equation 6.12).

$$f_{PA_i}(\Pi_i(\overrightarrow{t_i^{slk}})) = -c_i^{dl} \cdot dl_i(\Pi_i) - \sum_{\pi_{i,j,k}^P \in \Pi_i} rc(\pi_{i,j,k}^P) - \sum_{j=1}^{n_i} w_{i,j}(E(s_{i,j}^*) - s_{i,j}), \quad (6.12)$$

, where $c_i^{dl} \cdot dl_i(\Pi_i)$ is the delay cost, $\sum_{\pi_{i,j,k}^P \in \Pi_i} rc(\pi_{i,j,k}^P)$ is the resource cost, and $\sum_{j=1}^{n_i} w_{i,j}(E(s_{i,j}^*) - s_{i,j})$ is the instability cost.

In case the fitness value of a new individual is greater than an earlier obtained fitness value ($f_{PA_i}(\Pi_i(\overrightarrow{t_i^{slk,t}})) \leq f_{PA_i}(\Pi_i(\overrightarrow{t_i^{slk,t+1}}))$), the old individual is replaced by the new one. Otherwise, the old one remains.

D: Initialisation and Termination

The ES used by a project agent starts by computing an initial population, i.e., the first generation. In the (1,1)-ES, the first generation is also the first individual since the population size is 1. We let the ES start with a positive real-valued vector $\overrightarrow{\alpha}^0 = \langle \alpha_{i,1}^0, \dots, \alpha_{i,n_i}^0 \rangle$, where $\alpha_{i,j}^0$ is randomly chosen $0 \leq \alpha_{i,j}^0 \leq 1$.

As to the termination condition, we specify maximum number of generations for different experimental settings. Details of the experiments can be found in Section 6.3.

6.3 Experiments

In this section, we conduct experiments and empirically evaluate whether our proposed algorithms for the two types of agents produce efficient and robust schedules in a nondeterministic environment. We first describe our experimental setup (§6.3.1). The experimental results and analysis are presented in §6.3.2.

6.3.1 Experimental Setup

The problem instances that are used to evaluate our agent-based stable proactive scheduling algorithms are the same problem instances (i.e., 80 MPSPLib instances and 10 simulated AGH instances) as being used in Section 5.3.

The nondeterminism assumes that the uncertainty resides in the activity durations. In order to simulate the nondeterministic aspect of scheduling-environment uncertainty, we employ a random generator that generates stochastic actual activity processing durations. Project management literature suggests that activity durations generally follow a *beta distribution* (see Kerzner, 2006). The probability density function of a beta distribution is shown in Equation 6.13.

$$f(x, \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}, \quad (6.13)$$

where $B(\alpha, \beta)$ is a *beta function*: $B(\alpha, \beta) = \int_1^0 t^{\alpha-1}(1-t)^{\beta-1} dt$.

In the experiments, we have chosen the following parameters $\alpha = 2$ and $\beta = 5$ for generating the random numbers x : $f = (x, 2, 5)$. The chosen α and β values make the probability of x with a right-skewed beta distribution. It resembles the fact that the actual activity processing times are close to the estimated ones. Figure 6.14 plots the probability density function of the chosen beta distribution f . The minimum and maximum values of this distribution are chosen to be 0.5 times and 2.25 times of the estimated activity

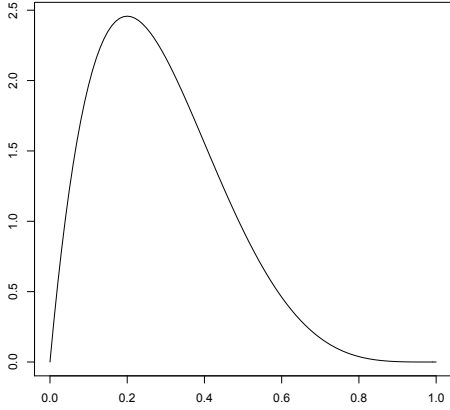


Figure 6.14: Probability density function of a beta ($\alpha = 2, \beta = 5$) distribution

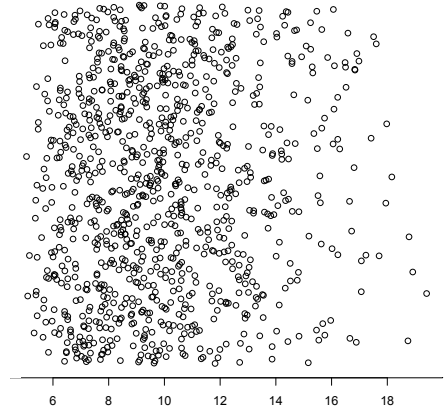


Figure 6.15: 1000 samples of actual activity processing duration $p_{i,j}^*$ ($p_{i,j} = 10$)

duration $p_{i,j}$. Thus, to simulate the actual activity durations $p_{i,j}^*$ we have the following equation (Equation 6.14).

$$p_{i,j}^* = \left(\frac{7}{4}x + \frac{1}{2}\right) \cdot p_{i,j} \quad (6.14)$$

When x equals to the expected value ($\frac{\alpha}{\alpha+\beta} = \frac{2}{7}$) of the beta distribution, the simulated activity processing duration equals to the estimated duration ($p_{i,j}^* = p_{i,j}$). Figure 6.15 shows 1000 samples of simulated actual activity processing durations $p_{i,j}^*$ when $p_{i,j} = 10$.

In the experiments, we assume (1) in case that the actual processing duration $p_{i,j}^*$ of an activity $a_{i,j}$ is smaller than the estimated one $p_{i,j}$ ($p_{i,j}^* < p_{i,j}$), unused resources will not be reallocated, (2) the weight factor $w_{i,j}$ in the stability measurement (Equation 6.7) is identical for all activities: $w_{i,j} = 20$, (3) project agents choose the same project unit delay cost ($c_i^d = 100, \forall P_i \in \mathcal{P}$), and (4) resource agents choose the same resource-unit-utilisation cost ($c_k^u = 1, \forall R_k \in \mathcal{R}$).

6.3.2 Results and Analysis

In this subsection, we present our experimental results for two distinct cases: (a) the performance of the three constructive heuristics (i.e., EFPF, RPF, and CPF) used by resource agents in improving schedule stability, and (b) the performance of the (1,1)-ES learning approach used by project agents for increasing their utilities. We recall that project-agent utility is a tradeoff between schedule efficiency and schedule stability (see §6.2.2).

A: Performance Evaluation of EFPF, RPF, and CPF

To evaluate the performance of the three constructive heuristics, 100 simulation runs for each problem instance have been generated. Table 6.1 shows the average stability of the

schedules in which the resource flows are organised by four different approaches. First, column 2 shows the average schedule stability where the resource flows are organised by Artigues et al. (2003). Since the Artigues et al. (2003) approach does not attempt to maximise schedule stability, we use the obtained stability as a benchmark. Column 3 to 5 shows the average schedule stability by using three proposed constructive heuristic approaches (EFPP, RPF, and CPF).

Of the three heuristics developed in the thesis, RPF performs generally the best. EFPP performs rather close to RPF, and the obtained stability is only slightly worse. CPF follows EFPP and it is able to improve the schedule stability from 2.2% to 14.9%.

Problem instances	Total project stability ($w_{i,j} = 20, \forall i \in \{1, \dots, m\} \wedge \forall j \in \{1, \dots, n_i\}$)			
	Artigues et al.	EFPP	RPF	CPF
I90/2	13064	10646 (-16.2%)	10504 (-17.3%)	12380 (-2.6%)
I90/5	18688	15862 (-15.1%)	15402 (-17.6%)	17252 (-7.7%)
I90/10	25724	21276 (-16.8%)	20702 (-19.5%)	22716 (-11.7%)
I90/20	36850	31658 (-14.1%)	30432 (-17.4%)	32244 (-12.5%)
I120/2	20682	17470 (-15.5%)	17326 (-16.2%)	19962 (-3.5%)
I120/5	31646	27282 (-13.8%)	26256 (-17.0%)	30040 (-5.1%)
I120/10	47880	39052 (-18.4%)	39060 (-18.4%)	42789 (-10.6%)
I120/20	83050	67384 (-18.9%)	65962 (-20.6%)	72408 (-12.8%)
IA2	1840	1586 (-13.8%)	1514 (-17.7%)	1800 (-2.2%)
IA3	2814	2401 (-14.7%)	2415 (-14.2%)	2698 (-4.1%)
IA6	5648	4815 (-14.7%)	4698 (-16.8%)	5159 (-8.7%)
IA10	9357	8227 (-12.1%)	8225 (-12.1%)	8307 (-11.2%)
IA20	18524	15749 (-15.0%)	15729 (-15.1%)	16161 (-12.8%)
IA30	29157	24282 (-16.7%)	23689 (-18.8%)	25389 (-12.9%)
IA40	39641	31547 (-20.4%)	31547 (-20.4%)	34087 (-14.0%)
IA60	65414	56048 (-14.3%)	55041 (-15.9%)	56154 (-14.2%)
IA80	78454	65848 (-16.1%)	64124 (-18.3%)	66887 (-14.7%)
IA120	125710	103415 (-17.7%)	102485 (-18.5%)	106940 (-14.9%)

Table 6.1: Comparison of three heuristics on total project stability over 100 simulations

From the results shown in Table 6.1, we observe that the improvements made by RPF and EFPP are insensitive to the project numbers in a problem instance and to the activity numbers in a project. In contrast, the performance of CPF is highly dependent on the number of projects in a problem instance. The more projects there are in a problem instance, the better CPF performs. However, for large problem instances it cannot outperform the other two heuristics.

In summary, we may conclude that the stability of a baseline schedule can be increased through proper resource allocation. From the three proposed heuristics, RPF and EFPP are preferred in general cases, and CPF is applicable when the problem size is large.

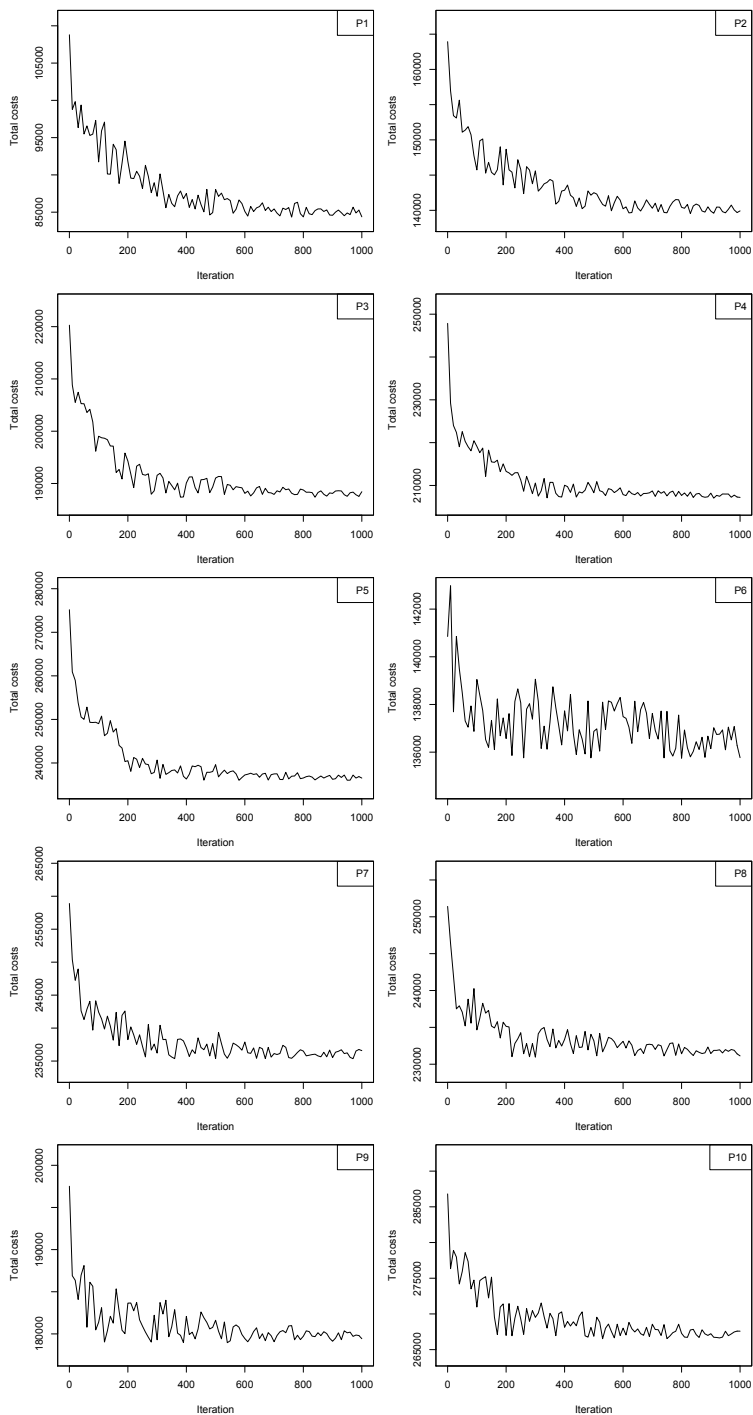


Figure 6.16: (1,1)-ES learning curves of the 10 projects in I90/10/1 with a particular instance of incidents

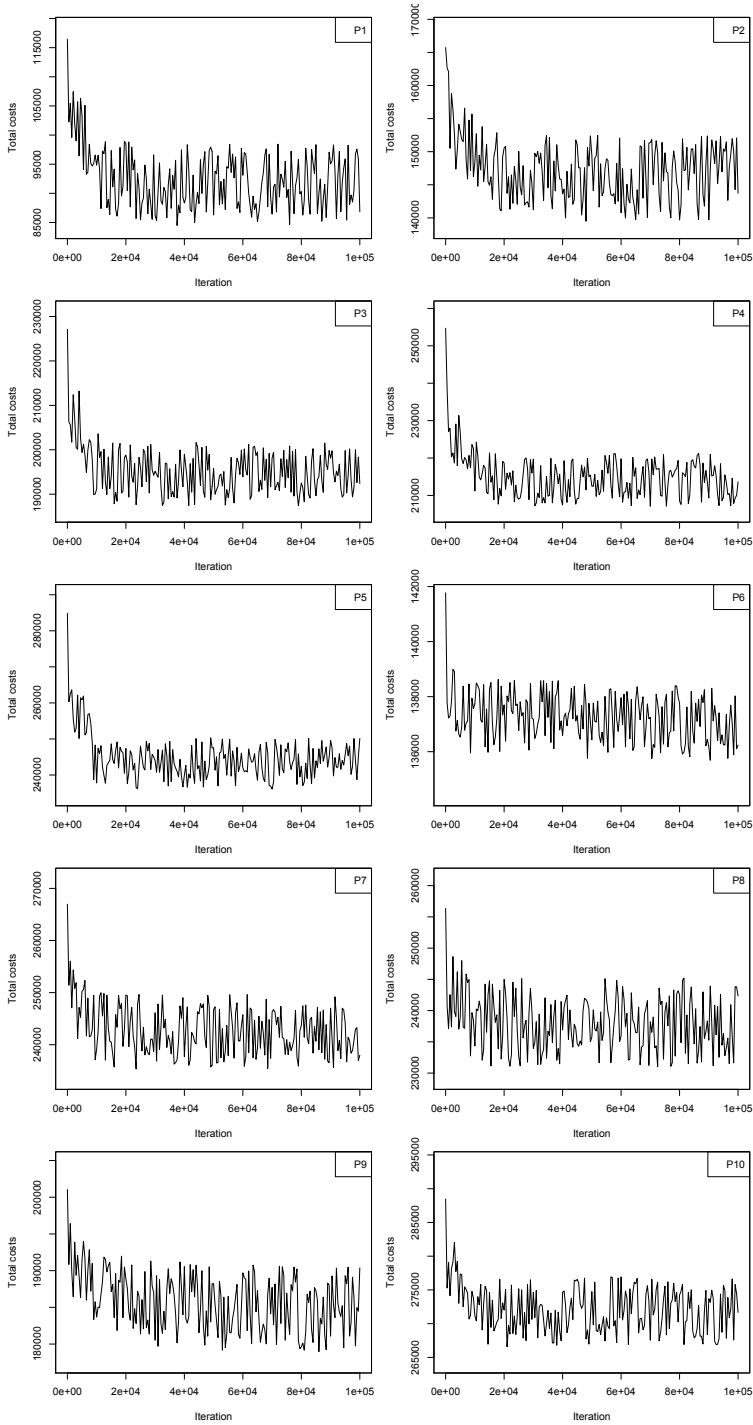


Figure 6.17: (1,1)-ES learning curves of the 10 projects in I90/10/1 with random instances of incidents

B: Performance Evaluation of (1,1)-ES

To evaluate the performance of the (1,1)-ES learning approach used by project agents, we consider two scenarios. First, we use the beta random generator to generate a particular instance of incidents. We let project agents learn a vector of slack time windows that maximise their utilities given that particular instance of incidents. Second, for every simulation, we generate new beta random incidents. Project agents are expected to learn a vector of slack time windows that perform well, in general, in maximising their utilities. The learnt vector is expected to improve schedule stability while keeping the schedule efficient (in terms of APD and TSRU).

To illustrate the results, we plot the learning curves of the 10 projects in instances I90/10/1 for the two scenarios in Figure 6.16 and Figure 6.17, respectively. For the particular-instance-incident scenario (see Figure 6.16), we see that (1,1)-ES can quickly (in average within 1000 iterations) learn a proper vector of slack time windows. However, for the random-instance-incident scenario (see Figure 6.17), the learning is rather slow. Project agents are able to learn a vector of slack time windows that improves their utilities to a certain extent. However, the learnt vector is not always the best.

In real-world project-scheduling environment, e.g., the one of AGH, one can never expect that a particular instance of incidents reoccurs in the same way as it did before. Yet, the good news is that incidents are never completely random. Based on certain environmental conditions (weather forecast, time schedule, historical data), a disruptive pattern can be predicted. Once a disruptive pattern is known, the proposed (1,1)-ES is able to learn quickly a proper vector of slack time windows that increases the stability of schedules, and safeguards the efficiency of schedules.

6.4 Answer to Research Question 3

In this chapter we addressed the third research question. [RQ3: *How can agents make and coordinate their local decisions in order to achieve a globally efficient and robust schedule in a nondeterministic environment?*] The notion of nondeterminism in a project-scheduling environment is represented as variable activity processing times.

Our research was as follows. First, we studied stability, i.e., a solution robustness often considered in project-scheduling problems. Subsequently, we proposed an agent-based stable proactive scheduling procedure. The procedure aimed at constructing an efficient and robust multi-project schedule under variable-activity-processing-time uncertainty. The procedure has two components: (1) three constructive scheduling heuristics employed by resource agents, and (2) a coevolving slack-time-window inserting strategy employed by project agents.

Based on the experiments and the analyses given above, we may conclude that the proposed stable proactive scheduling procedure is effective in constructing both efficient and robust schedules, therefore answer our RQ3.

Chapter 7

Conclusions

In this chapter, we provide a conclusive answer to our research questions and to the problem statement. In Section 7.1, we repeat the three research questions and summarise the answers given in earlier chapters. Based on the answers to the research questions, Section 7.2 formulates our answer to the problem statement. Finally, in Section 7.3, we discuss our research in a broader perspective by giving recommendations for future research.

7.1 Answers to the Research Questions

In Section 1.2, we formulated three research questions. Throughout the thesis, we addressed and answered these questions in three consecutive chapters (Chapter 4, 5, and 6). In this section, the answers to the three research questions are revisited and placed in an overall context.

7.1.1 Agent-based Model for AGH Scheduling Problem

As we have shown in Section 1.1, current AGH scheduling problems are decision problems with large-scale informational and managerial decentralisation. The scheduling process is a distributed decision process amongst multiple organisations. Multiagent systems have proved to be suitable to address such distributed decision-making problems. Employing a multiagent system as our solution framework calls for an adequate agent-based model of the AGH scheduling problem. This led to our first research question.

RQ1: *How can an AGH scheduling problem be represented in an agent-based model?*

In Chapter 4, we showed that the AGH scheduling problem can be effectively modelled by an agent-based model. The proposed model consists of (1) two classes of role-based agents — resource agents and project agents — and (2) a lease-based market mechanism for coordinating the autonomous scheduling decisions of the individual agents.

For modelling agents, we adopted a physical-entity-oriented modelling approach. The chosen physical-entity-oriented approach provides a natural description of the AGH domain by incorporating the two behavioural entities: (i) ground-handling service providers (resource agents), and (ii) aircraft ground-handling managers (project agents). For modelling project agents, we chose to use a coarse-grained approach. The chosen coarse-grained approach allows the modelling of self-interested agents and avoids inter-agent communicational overload. The self-interested nature of the agents is represented by the utility modelling of each of the two classes of agents. The chosen agent representation offers properties such as self-interestedness and scalability to the agent-based scheduling system.

For modelling agent interactions, we proposed a lease-based market mechanism that coordinates the scheduling decisions among two classes of heterogeneous agents. In the mechanism, we introduced a concept of resource-time slot, that is used as a common language for inter-agent interactions. For processing an activity, a lease (i.e., a time-resource slot) is negotiated by a resource agent and a project agent. The agents evaluate the value of the slot based on their own value systems (marginal agent utilities). As a result, the proposed coordination mechanism successfully distributes the scheduling decisions over autonomous decision makers. As long as the slot chosen makes both the resource-agent schedule and the project-agent schedule feasible, a global feasible schedule will be obtained. The proposed coordination mechanism offers properties such as flexibility and scalability to the agent-based scheduling system.

7.1.2 Efficiency and Robustness under Partial Observability

The AGH scheduling environment is well known for its large number of disturbances stemming from various sources. The disturbances cause a high degree of uncertainty in making AGH schedules. In the thesis, we first focused on investigating scheduling solutions under partial observability. Partial observability in AGH scheduling can be interpreted as variable aircraft arrival times, i.e., the actual arrival time of an aircraft at the airport is often different from (most of the times, it is later than) the one foreseen in the original flight plan. This led to our second research question.

RQ2: *How can agents make and coordinate their local decisions in order to achieve a globally efficient and robust schedule in a partially observable environment?*

In Chapter 5 we proposed OI-MAS, an online iterative multiagent scheduling approach. OI-MAS deals with the efficiency and the robustness of AGH schedules by two components: (1) a clairvoyant online schedule generation scheme (COSGS), and (2) an iterative schedule-improvement method (ISIM).

Within the two components, the COSGS focuses on the robustness aspect. In the COSGS, project agents adopt a clairvoyant online scheduling scheme in which the scheduling process of a project agent starts as soon as the project is **actually** released. Employing the COSGS effectively eliminates the possibility of schedule disruptions caused by the variable project release times. Evidently, the schedules obtained are robust with respect to the the variable project release times.

Furthermore, the second component ISIM of OI-MAS focuses on efficiency. In the ISIM, project agents and resource agents revise iteratively their earlier determined schedules in order to increase their utilities. Based on the experiments and the analyses, we may conclude that the ISIM is effective, and the schedules constructed by OI-MAS are efficient.

In summary, we may conclude that OI-MAS provides efficient and robust AGH schedules in a partially observable environment.

7.1.3 Efficiency and Robustness under Nondeterminism

Apart from partial observability, the second class of uncertainty we have investigated in this thesis is nondeterminism. Nondeterminism in AGH scheduling context is represented by variable ground-handling operational times. During the scheduling phase of AGH, each of the ground-handling operations has an estimated processing duration. However, during the execution phase, the actual processing durations may differ from the original estimations. Nondeterminism led to our third research question.

RQ3: *How can agents make and coordinate their local decisions in order to achieve a globally efficient and robust schedule in a nondeterministic environment?*

In Chapter 6 we studied stability, i.e., a solution robustness often considered in project-scheduling problems. Subsequently, we proposed an agent-based stable proactive scheduling procedure. The procedure aims at constructing an efficient and robust multi-project schedule in a nondeterministic environment.

The proposed agent-based stable scheduling procedure has two components: (1) three constructive scheduling heuristics employed by resource agents, and (2) a coevolving slack-time-window inserting strategy employed by project agents. Based on the experiments and the analyses given in Section 6.3, we may conclude that the proposed stable proactive scheduling procedure is effective in constructing both efficient and robust schedules. Therefore, we answered our RQ3.

7.2 Answer to the Problem Statement

In this section, we provide an answer to the problem statement posed in Chapter 1. Our answer is based on the answers to the three research questions discussed in the previous section. First, let us reiterate our problem statement.

PS: *Can a number of self-interested agents, by coordinating their local scheduling decisions, achieve a global AGH schedule that is both efficient and robust?*

Our research provides an *affirmative* answer to the problem statement, of which the essence may be summarised in three parts. First, we proposed an agent-based model for the AGH scheduling problem. In the proposed model, two classes of role-based self-interested agents (resource agents and project agents) coordinate their local decisions in

a lease-based market mechanism. The model provides three desired properties that are self-interestedness, flexibility, and scalability. Second, we addressed schedule efficiency and robustness in a partial observable environment. The proposed OI-MAS approach dealt with efficiency by employing an iterative schedule improvement method and dealt with robustness by a clairvoyant online schedule generation scheme. Third, we addressed schedule efficiency and robustness in a nondeterministic environment. Three constructive heuristic approaches were proposed for resource agents to increase the schedule robustness. A (co)evolutionary-strategy-based slack-time-window inserting approach was proposed for project agents to increase the schedule efficiency and robustness.

7.3 Recommendations for Future Research

The research presented in the thesis indicates several important and promising areas of future research. In this section, we mention three of the most interesting areas.

Generalisation to various real-world scheduling problems

In our research, the AGH scheduling problem is formulated as a DRCMPSP/u. It is a fairly generalised and realistic multi-project scheduling problem. Yet, many other interesting generalisations of real-world project scheduling problems are worth to be further investigated. Below, we mention two of them.

1. *Generalisation of project-deadline constraints.* An assumption we made for AGH scheduling problems is that aircraft turnaround process can be delayed ‘forever’ until a feasible schedule is found. In practice, a turnaround process has often a hard deadline. Once a deadline constraint is imposed, the primary scheduling objective is to find a **feasible** schedule. It is interesting to investigate how the agents in our MAS scheduling system can construct feasible schedules under project-deadline constraints (or even under variable project-deadline constraints).
2. *Generalisation of resource transition.* The assumption to be relaxed is that the resource transition periods are irrelevant. In the thesis, we assumed that resources released from an activity can be immediately used by another activity. In practice, there is often a transition period needed to transfer resources from one activity to another. Transition time may vary according to the resource type and the distance between the two activities. A generalisation of our agent-based scheduling model would consider the transition period during the scheduling process.

Cooperation in agent negotiations

In our research we assumed complete self-interestedness of all agents, and pure competition in the sense that the agents try to maximise their own utilities in every round of inter-agent negotiation. Research in game theory and social science has shown that self-interestedness and mutual aid are not at all incompatible (see, e.g., Axelrod, 1984; Mayoh, 2002). Cooperative behaviour in self-interested multiagent games shows promising results. For instance, cooperation in non-zero games increase the overall social welfare as well as

the individual utilities (see de Jong, 2009). Therefore, an interesting direction of future research is to investigate cooperative behaviours of the project agents and the resource agents in the proposed MAS scheduling system. For instance, the following two research topics can be addressed: (1) how can agents behave cooperatively in the proposed market-based coordination mechanism? and (2) what are the benefits of introducing cooperation in scheduling?

Solid agent learning research

In the second class of scheduling problems under uncertainty, we have empirically investigated a learning approach (i.e., (1,1)-ES) for constructing a stable project schedule by inserting appropriate inter-activity slack time windows. Abstracting from the application domain of AGH, we may state that the issue is to learn an optimal policy mapping from a continuous domain of states to a continuous domain of actions based on observations about incidents. Here, solid agent learning research can be conducted to investigate the applicability of various learning approaches, both theoretically and empirically. These approaches, for instance, include reinforcement learning (see Roos (2010) for stochastic approximation on robust scheduling) and Nash-Q learning (to be transferred to this domain).

References

- ACI (2009). ACI Annual World Airport Traffic Report 2009. Technical report, Airports Council International. [79]
- Aggoun, A. and Beldiceanu, N. (1993). Extending CHIP in order to solve complex scheduling and placement problems. *Mathematical and computer modelling*, 17(7):57–73. [36]
- Airport Research Center (2009). Study on the impact of directive 96/67/ec on ground handling services 1996-2007. Technical report, Airport Research Center, Aachen, Germany. [4]
- Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843. [xvii, 12, 13]
- Araújo, J. A. A., Pavón, J., López-Paredes, A., and Pajares, J. (2009). Agent-based modeling and simulation of multi-project scheduling. In Baldoni, M., Baroglio, C., Bentahar, J., Boella, G., Cossentino, M., Dastani, M., Dunin-Keplicz, B., Fortino, G., Gleizes, M. P., Leite, J., Mascardi, V., Padget, J. A., Pavón, J., Polleres, A., Fallah-Seghrouchni, A. E., Torroni, P., and Verbrugge, R., editors, *Proceedings of the 2nd Multi-Agent Logics, Languages, and Organisations Federated Workshops, MALLOW 2009*, volume 494, Turin, Italy. [39]
- Artigues, C., Michelon, P., and Reusser, S. (2003). Insertion techniques for static and dynamic resource constrained project scheduling. *European Journal of Operational Research*, 149:249–267. [91, 106]
- Axelrod, R. M. (1984). *The Evolution of Cooperation*. Basic Books, New York, NY, USA, illustrated, revised edition. [114]
- Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, USA, 1st edition. [103]
- Beni, G. and Wang, J. (1989). Swarm intelligence in cellular robotic systems. In *Proceedings of NATO Advanced Workshop on Robots and Biological Systems*, Tuscany, Italy. [36]

- Błażewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., and Weglarz, J. (2007). *Handbook on Scheduling: From Theory to Applications*. Springer-Verlag, Berlin-Heidelberg-New York, 1st edition. [5]
- Błażewicz, J., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1983). Scheduling subject to resource constraints: Classification and complexity. *Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 5:11–24. [17, 32]
- Boer, K., Kayamak, U., and Spiering, J. (2007). From discrete-time models to continuous-time, asynchronous modeling of financial markets. *Computational Intelligence*, 23:142–161. [42]
- Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. In *Proceedings of the National Academy of Sciences of the United States of American (PNAS)*, volume 99, pages 7280–7287. [47]
- Bonfill-Teixidor, A. (2006). *Proactive management of uncertainty to improve scheduling robustness in process industries*. PhD thesis, Departament d’Enginyeria Quimica, Escola Tecnica Superior d’Enginyeria Industrial de Barcelona, Universitat Politecnica de Catalunya, Spain. [25, 43]
- Bouleimen, K. and Lecocq, H. (2000). Multi-objective simulated annealing for the resource-constrained multi-project scheduling problem. In *Proceedings of the 7th International Workshop on Project Management and Scheduling (PMS 2000)*, Osnabrück, Germany. [36]
- Browning, T. R. and Yassine, A. A. (2010). Resource-constrained multi-project scheduling: Priority rule performance revisited. *International Journal of Production Economics*, 126(2):212–228. [32, 33, 34, 35]
- Brucker, P. (2003). *Scheduling Algorithms*. Springer-Verlag, Berlin-Heidelberg-New York, 4th edition. [5]
- Cesta, A., Oddi, A., and Smith, S. F. (2002). A constraint-based method for project scheduling with time windows. *Journal of Heuristics*, 8(1):109–136. [36]
- Confessore, G., Giordani, S., and Rismondo, S. (2007). A market-based multi-agent system model for decentralized multi-project scheduling. *Annals of Operations Research*, 150:115–135. [5, 22, 24, 25, 38, 39, 40, 41, 42, 45, 47, 48, 52]
- Davenport, A. J. and Beck, J. C. (2000). A survey of techniques for scheduling with uncertainty. Unpublished, 2000. [94]
- Davenport, A. J., Gefflot, C., and Beck, J. C. (2001). Slack-based techniques for robust schedules. In *Proceedings of the 6th European Conference on Planning (ECP 2001)*, Toledo, Spain. [89, 93]
- de Jong, A. (2010). *Improving industrial maintenance contract relationships: optimising the allocation of decision rights and risks in contract relationships involving uncertainty of demand*. PhD thesis, Delft University of Technology, Delft, The Netherlands. [1]

-
- de Jong, S. (2009). *Fairness in Multi-Agent Systems*. PhD thesis, Department of Knowledge Engineering, Maastricht University, Maastricht, The Netherlands. [115]
- Deblaere, F., Demeulemeester, E., Herroelen, W., and Van de Vonder, S. (2007). Robust resource allocation decisions in resource-constrained projects. *Decision Sciences*, 38(1):5–37. [93, 95]
- Deckro, R. F., Winkofsky, E. P., Hebert, J. E., and Gagnon, R. (1991). A decomposition approach to multi-project scheduling. *European Journal of Operational Research*, 51(1):110–118. [31, 33]
- Demeulemeester, E. L. and Herroelen, W. (2002). *Project Scheduling: A Research Handbook*. Kluwer Academic Publishers, Boston-Dordrecht-London. [10, 11, 20, 29, 31, 43]
- Deng, L. and Lin, Y. (2007). A particle swarm optimization for resource-constrained multi-project scheduling problem. In *Proceedings of 2007 International Conference on Computational Intelligence and Security (CIS 2007)*, pages 1010–1014, Harbin, China. [36]
- Dorndorf, U. (2002). *Project Scheduling with Time Windows: From Theory to Applications*. Physica-Verlag, Heidelberg-New York. [9, 13, 14, 36]
- Durfee, E. H. and Rosenschein, J. S. (1994). Distributed problem solving and multi-agent systems: Comparisons and examples. In *Proceedings of the 13th International Workshop on Distributed Artificial Intelligence*, pages 52–62. [41]
- Elmaghraby, S. E. and Kamburowski, J. (1992). The analysis of activity networks under generalized precedence relations (GPRs). *Management Science*, 38(9):1245–1263. [14]
- EU Council (1996). Council Directive 96/67/EC of 15 October 1996 on access to the groundhandling market at Community airports. *Official Journal of the European Union*, L 272:36–45. [3, 127]
- EuroControl (2008). Challenges of growth 2008. Technical report, Air Traffic Statistics and Forecasts (STATFOR), EuroControl, Brussels, Belgium. [1]
- EuroControl (2010). Delays to Air Transport in Europe, Digest - Annual 2009. Technical report, Central Office for Delay Analysis (CODA), EuroControl, Brussels, Belgium. [2]
- Fendley, L. G. (1968). Toward the development of a complete multiproject scheduling system. *Journal of Industrial Engineering*, 19(10):505–515. [33]
- Gantt, H. L. (1974). *Work, Wages, and Profits*. Hive Publishing. Co., Easton, PA, USA, 2nd, illustrated edition. [2]
- Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, CA, USA. [32]
- Ghosh, S. (1996). *Guaranteeing fault tolerance through scheduling in real-time systems*. PhD thesis, University of Pittsburgh, Pittsburgh, PA, USA. [94]

- Gonçalves, J. F., , de Magalhães Mendes, J. J., and Resende, M. G. C. (2008). A genetic algorithm for the resource constrained multi-project scheduling problem. *European Journal of Operational Research*, 189(3):1171–1190. [35]
- Gonsalves, T., Ito, A., Kawabata, R., and Itoh, K. (2008). Swarm intelligence in the optimization of software development project schedule. In *Proceedings of the 32nd Annual IEEE International Computer Software and Applications Conference (COMPSAC 2008)*, pages 587–592, Washington, DC, USA. IEEE Computer Society. [36]
- Graham, B. and Guyer, C. (1999). Environmental sustainability, airport capacity and european air transport liberalization: irreconcilable goals? *Journal of Transport Geography*, 7(3):165–180. [1]
- Gualandi, N., Mantecchini, L., and Serrau, D. (2006). Environmental capacity and sustainability of european regional airports: A case study. In *Proceedings of World Academy of Science, Engineering and Technology (PWASET 2006)*, volume 17, pages 182–187, Paris, France. [1]
- Guldmond, T., Hurink, J., Paulus, J., and Schutten, J. (2008). Time-constrained project scheduling. *Journal of Scheduling*, 11(2):137–148. [16]
- Hall, N. G. and Posner, M. E. (2004). Sensitivity analysis for scheduling problem. *Journal of Scheduling*, 7(1):49–83. [44]
- 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(5):563–577. [16, 29]
- Hartmann, S. and Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1–14. [22]
- Herroelen, W. and Leus, R. (2004). The construction of stable project baseline schedules. *European Journal of Operational Research*, 156(3):550–565. [88]
- Herroelen, W. and Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 165(2):289–306. [44]
- Herroelen, W., Reyck, B. D., and Demeulemeester, E. (1998). Resource-constrained project scheduling: a survey of recent developments. *Computers and Operations Research*, 25(4):279–302. [31]
- Heydenreich, B., Müller, R., and Uetz, M. (2006). Decentralization and mechanism design for online machine scheduling. In Arge, L. and Freivalds, R., editors, *Proceedings of the 10th Scandinavian Workshop on Algorithm Theory, Riga, Latvia, July 6-8, 2006.*, number 4059/2006 in Lecture Notes in Computer Science. [38, 47]
- Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, Ann Arbor, MI, USA, illustrated edition. [35]

-
- 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. [5, 25, 38, 39, 40, 41, 42, 45, 47, 48, 52, 78]
- Hu, J. and Wellman, M. P. (2003). Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069. [101]
- IATA (2009). *IATA Airport Handling Manual*. International Air Transport Association, Montreal, Quebec, Canada, 29th edition. [2]
- Kerzner, H. (2006). *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*. John Wiley & Sons Inc., Hoboken, NJ, USA, 9th, illustrated, revised edition. [9, 104]
- Kim, K. W., Yun, Y., Yoon, J. M., Gen, M., and Yamazaki, G. (2005). Hybrid genetic algorithm with adaptive abilities for resource-constrained multiple project scheduling. *Computers in Industry*, 56(2):143–160. [35]
- Kirkpatrick, S., Gelatt, C. D., Jr., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671–680. [35]
- 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*, pages 147–178. Kluwer Academic Publishers. [32, 33, 69]
- Kolisch, R. and Sprecher, A. (1997). PSPLIB — a project scheduling problem library. *European Journal of Operational Research*, 96:205–216. [78]
- Krüger, D. and Scholl, A. (2007). A heuristic solution framework for the resource constrained multi-project scheduling problem with sequence-dependent transfer times. Jena Research Papers in Business and Economics 16/2007, Friedrich-Schiller-University Jena, School of Economics and Business Administration. [31]
- Kumanan, S., Jose, G. J., and Raja, K. (2006). Multi-project scheduling using an heuristic and a genetic algorithm. *The International Journal of Advanced Manufacturing Technology*, 31(3-4):360–366. [35]
- Kumar, V. (1992). Algorithms for constraint-satisfaction problems: A survey. *Artificial Intelligence Magazine*, 13(1):32–44. [36]
- Kurtulus, I. S. (1985). Multiproject scheduling: Analysis of scheduling strategies under unequal delay penalties. *Journal of Operations Management*, 5(3):291–307. [24]
- Kurtulus, I. S. and Davis, E. (1982). Multi-project scheduling: Categorization of heuristic rules performance. *Management Science*, 28(2):161–172. [32, 33, 34, 35]
- Kurtulus, I. S. and Narula, S. C. (1985). Multi-project scheduling: Analysis of project performance. *IIE Transactions on Engineering Management*, 17(1):58–66. [32]

- Lambrechts, O., Demeulemeester, E., and Herroelen, W. (2010). Time slack-based techniques for robust project scheduling subject to resource uncertainty. *Annals of Operations Research*. Published online: 29 August 2010. [95]
- Lee, Y.-H. (2002). *Market-based dynamic resource control of distributed multiple projects*. PhD thesis, The Pennsylvania State University, University Park, PA, USA. [38, 39, 40, 41, 42, 45, 48, 52]
- Lee, Y.-H., Kumara, S. R. T., and Chatterjee, K. (2003). Multiagent based dynamic resource scheduling for distributed multiple projects using a market mechanism. *Journal of Intelligence Manufacturing*, 14:471–484. [39, 47, 48]
- Leus, R. (2003). *The generation of stable project plans*. PhD thesis, Catholic University of Leuven, Leuven, Belgium. [88, 93, 95]
- Lewis, J. P. (2005). *Project Planning, Scheduling, and Control: A Hands-on Guide to Bringing Projects in on Time and on Budget*. McGraw-Hill Professional, New York, NY, USA, 4th, illustrated edition. [9]
- Lockyer, K. and Gordon, J. (2005). *Project Management and Project Network Techniques*. Prentice Hall, Harlow, Essex, England, 7th edition. [11]
- Lorterapong, P. and Rattanadamrongagsorn, T. (2001). Viewing construction scheduling as a constraint satisfaction problem. In *Proceedings of the 6th international conference on application of artificial intelligence to civil & structural engineering (ICAAICSE 2001)*, pages 19–20. Civil-Comp Press. [36]
- Lova, A., Maroto, C., and Tormos, P. (2000). A multicriteria heuristic method to improve resource allocation in multiproject scheduling. *European Journal of Operational Research*, 127(2):408–424. [32, 79]
- Lova, A. and Tormos, P. (2001). Analysis of scheduling schemes and heuristic rules performance in resource-constrained multiproject scheduling. *Annals of Operations Research*, 102(1-4):263–286. [29, 31, 32, 33, 34, 35, 79]
- Lova, A. and Tormos, P. (2002). Combining random sampling and backward-forward heuristics for resource-constrained multi-project scheduling. In *Proceedings of the 8th International Workshop on Project Management and Scheduling*, pages 244–248, Valencia, Spain. [32]
- Lova, A., Tormos, P., and Barber, F. (2006). Multi-mode resource constrained project scheduling: Scheduling schemes, priority rules and mode selection rules. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 30:69–86. [18]
- Mauroy, G., Wardi, Y., and Proth, J. (1997). Sensitivity analysis of priority-based scheduling in manufacturing. In *Proceedings of the 6th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 1997)*, pages 465–466, Los Angeles, CA, USA. [44]

-
- Mayoh, B. H. (2002). Evolution of cooperation in multiagent systems. In *Proceedings of the 1st EurAsian Conference on Information and Communication Technology*, volume 2510 of *Lecture Notes in Computer Science*, pages 701–710, London, UK. Springer-Verlag. [114]
- Neumann, K., Schwindt, C., and Zimmermann, J. (2001). *Project Scheduling with Time Windows and Scarce Resources*. Springer-Verlag, Berlin-Heidelberg-New York, 2nd edition. [11, 16, 20, 31]
- Neumann, K. and Zimmermann, J. (1999). Resource levelling for projects with schedule-dependent time windows. *European Journal of Operational Research*, 117(3):591–605. [10]
- Nuijten, W. and Le Pape, C. (1998). Constraint-based job shop scheduling with Ilog scheduler. *Journal of Heuristics*, 3(4):271–286. [36]
- Nuijten, W. P. M. (1994). *Time and resource constrained scheduling: a constraint satisfaction approach*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands. [5, 36]
- Oliehoek, F. A. (2010). *Value-based Planning for Teams of agents in Stochastic Partially Observable Environments*. PhD thesis, University of Amsterdam, Amsterdam, the Netherlands. [26]
- Özdamar, L. and Ulusoy, G. (1995). A survey on the resource-constrained project scheduling problem. *IIE Transactions*, 27(5):574–586. [31]
- Paredis, J. (2000). Coevolutionary algorithms. In Bäck, T., Fogel, D. B., and Michalewicz, Z., editors, *Evolutionary Computation 2: Advanced Algorithms and Operators*, chapter 23, pages 224–238. Institute of Physics Publishing, Bristol, UK. [101]
- Payne, J. H. (1995). Management of multiple simultaneous projects: a state-of-the-art review. *International Journal of Project Management*, 13(3):163–168. [22]
- Pennypacker, J. S. and Dye, L. D., editors (2002). *Managing Multiple Projects: Planning, Scheduling, and Allocating Resources for Competitive Advantage*. CRC Press, New York, NY, USA, illustrated edition. [22]
- Pritsker, A. A. B., Watters, L. J., and Wolfe, P. M. (1969). Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science*, 16(1):93–108. [31]
- Ran, Y.-P. (2003). *Repair-Based Scheduling*. PhD thesis, Institute for Knowledge and Agent Technology, Maastricht University, Maastricht, The Netherlands. [36]
- Roos, N. (2010). Cost optimal robust-schedules: Policy search in continuous action space using stochastic feedback. In *Proceedings of the 22nd Benelux Conference on Artificial Intelligence*. [115]

- Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Pearson Education, Inc., Upper Saddle River, NJ, USA, 2nd edition. [5, 37]
- Sadeh, N., Otsuka, S., and Schelback, R. (1993). Predictive and reactive scheduling with the microboss production scheduling and control system. In *Proceedings of the IJCAI-93 workshop on knowledge-based production planning, scheduling and control*, pages 293–306. [90]
- Schmidbergera, S., Balsb, L., Hartmann, E., and Jahns, C. (2009). Ground handling services at european hub airports: Development of a performance measurement system for benchmarking. *International Journal of Production Economics*, 117(1):104–116. [21]
- Servakh, V. V. and Shcherbinina, T. A. (2007). Complexity of project scheduling problem with nonrenewable resources. In Kalcsics, J. and Nickel, S., editors, *Operations Research Proceedings 2007*, pages 427–431, Berlin-Heidelberg. Springer-Verlag. [17]
- Sevaux, M. and Sörensen, K. (2002). A genetic algorithm for robust schedules in a just-in-time environment. Working Papers 2003008, Faculty of Applied Economics, University of Antwerp, Belgium. [87]
- Shankar, V. and Nagi, R. (1996). A flexible optimization approach for multi-resource, multi-project planning and scheduling. In *Proceedings of the 5th Industrial Engineering Research Conference*, pages 263–267, Minneapolis, MN, USA. [36]
- Shen, W. and Norrie, D. H. (1999). Agent-based systems for intelligent manufacturing: A state-of-the-art survey. *International Journal of Knowledge and Information Systems*, 1(2):129–156. [48]
- Shoham, Y. and Leyton-Brown, K. (2009). *Multiagent Systems*. Cambridge University Press, New York, NY, USA. [38, 47]
- Słowiński, R. and Hapke, M., editors (1999). *Scheduling Under Fuzziness - Studies in Fuzziness and Soft Computing*. Physica-Verlag, Heidelberg, illustrated edition. [43]
- Słowiński, R., Soniewicki, B., and Weglarz, J. (1994). DSS for multiobjective project scheduling. *European Journal of Operational Research*, 79(2):220–229. [21]
- ter Mors, A. W. (2010). *The world according to MARP: Multi-agent Route Planning*. PhD thesis, Delft University of Technology, Delft, The Netherlands. [1]
- T’kindt, V. and Billaut, J.-C. (2006). *Multicriteria scheduling: theory, models and algorithms*. Springer, Berlin-Heidelberg-New York, 2nd, illustrated edition. [21]
- Tobis, M. and Tobis, I. (2002). *Managing Multiple Projects*. McGraw-Hill Professional, New York, NY, USA, 1st edition. [22]
- Tsang, E. P. K. (1993). *Foundations of Constraint Satisfaction*. Academic Press, London and San Diego. [36]

-
- Turner, J. R. (2008). *The Handbook of Project-Based Management*. McGraw-Hill Professional, Glasgow, UK, 3rd, illustrated edition. [22]
- van de Vonder, S., Demeulemeester, E., and Herroelen, W. (2007). A classification of predictive-reactive project scheduling procedures. *Journal of Scheduling*, 10(3):195–207. [43, 90]
- van Leeuwen, P. and Witteveen, C. (2009). Temporal decoupling and determining resource needs of autonomous agents in the airport turnaround process. In Lang, J., editor, *IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies (WI-IAT 2009)*, pages 185–192, Los Alamitos, USA. IEEE Computer Society. [2, 22]
- Vercellis, C. (1994). Constrained multi-project plannings problems: A lagrangean decomposition approach. *European Journal of Operational Research*, 78(2):267–275. [31]
- Vestjens, A. P. (1997). *On-Line Machine Scheduling*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands. [68]
- Vila, L. (1994). A survey on temporal reasoning in artificial intelligence. *AI Communications*, 7(1):4–28. [12]
- Wallace, S. W. (2000). Decision making under uncertainty: is sensitivity analysis of any use? *Operations Research*, 48(1):20–25. [44]
- Wauters, T., Verbeeck, K., Vanden Berghe, G., and De Causmaecker, P. (2010). A game theoretic approach to decentralized multi-project scheduling (extended abstract). In Van der Hoek, Kaminka, Lespérance, Luck, and Sen, editors, *Proceedings of 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 1415–1416, Toronto, Canada. [5, 38, 39, 40, 41, 42, 45, 47, 48, 52, 80]
- Weibull, J. W. (1997). *Evolutionary Game Theory*. The M.I.T. Press, Cambridge, MA, USA. [101]
- Weiss, G., editor (2000). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, London, UK, illustrated edition. [38, 41]
- Wellman, M. P., Walsh, W. E., Wurman, P. R., and MacKie-Mason, J. K. (2001). Auction protocols for decentralized scheduling. *Games and Economic Behavior*, 35(1-2):271–303. [5, 38]
- Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. John Wiley & Sons, West Sussex, UK, 2nd edition. [38]
- Xue, F. and Fan, W. (2007). Multi-agent optimization design for multi-resource job shop scheduling problems. In *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, volume 4682/2007 of *Lecture Notes in Computer Science*, pages 1193–1204. [22]

- Yassine, A. A., Meier, C., and Browning, T. R. (2007). Multi-project scheduling using competent genetic algorithms. Working paper, Department of Industrial & Enterprise Systems Engineering, University of Illinois. PDL-2007-01. [35]

Appendix A

Airport Ground-Handling Operations

The following list extracted from European Council Directive 96/67/EC (EU Council, 1996) provides an exhaustive range of operations that ground handlers deal with for common commercial flights.

1. **Ground administration and supervision** comprise:
 - (a) representation and liaison services with local authorities or any other entity, disbursements on behalf of the airport user and provision of office space for its representatives;
 - (b) load control, messaging and telecommunications;
 - (c) handling, storage and administration of unit load devices;
 - (d) any other supervision services before, during or after the flight and any other administrative service requested by the airport user.
2. **Passenger handling** comprises any kind of assistance to arriving, departing, transfer or transit passengers, including checking tickets and travel documents, registering baggage and carrying it to the sorting area.
3. **Baggage handling** comprises handling baggage in the sorting area, sorting it, preparing it for departure, loading it on to and unloading it from the devices designed to move it from the aircraft to the sorting area and vice versa, as well as transporting baggage from the sorting area to the reclaim area.
4. **Freight and mail handling** comprises:
 - (a) for freight: physical handling of export, transfer and import freight, handling of related documents, customs procedures and implementation of any security procedure agreed between the parties or required by the circumstances;

- (b) for mail: physical handling of incoming and outgoing mail, handling of related documents and implementation of any security procedure agreed between the parties or required by the circumstances.

5. **Ramp handling** comprises:

- (a) marshalling the aircraft on the ground at arrival and departure;
- (b) assistance to aircraft packing and provision of suitable devices;
- (c) communication between the aircraft and the air-side supplier of services;
- (d) the loading and unloading of the aircraft, including the provision and operation of suitable means, as well as the transport of crew and passengers between the aircraft and the terminal, and baggage transport between the aircraft and the terminal;
- (e) the provision and operation of appropriate units for engine starting;
- (f) the moving of the aircraft at arrival and departure, as well as the provision and operation of suitable devices;
- (g) the transport, loading on to and unloading from the aircraft of food and beverages.

6. **Aircraft services** comprise:

- (a) the external and internal cleaning of the aircraft, and the toilet and water services;
- (b) the cooling and heating of the cabin, the removal of snow and ice, the de-icing of the aircraft;
- (c) the rearrangement of the cabin with suitable cabin equipment, the storage of this equipment.

7. **Fuel and oil handling** comprises:

- (a) the organisation and execution of fuelling and defuelling operations, including the storage of fuel and the control of the quality and quantity of fuel deliveries;
- (b) the replenishing of oil and other fluids.

8. **Aircraft maintenance** comprises:

- (a) routine services performed before flight;
- (b) non-routine services requested by the airport user;
- (c) the provision and administration of spare parts and suitable equipment;
- (d) the request for or reservation of a suitable parking and/or hangar space.

9. **Flight operations and crew administration** comprise:

- (a) preparation of the flight at the departure airport or at any other point;
- (b) in-flight assistance, including redispatching if needed;

- (c) post-flight activities;
- (d) crew administration.

10. **Surface transport** comprises:

- (a) the organisation and execution of crew, passenger, baggage, freight and mail transport between different terminals of the same airport, but excluding the same transport between the aircraft and any other point within the perimeter of the same airport;
- (b) any special transport requested by the airport user.

11. **Catering services** comprise:

- (a) liaison with suppliers and administrative management;
- (b) storage of food and beverages and of the equipment needed for their preparation;
- (c) cleaning of this equipment;
- (d) preparation and delivery of equipment as well as of bar and food supplies.

Appendix B

Properties of the 80 Chosen MPSPLib Instances

In this appendix, we provide an overview of the properties of all 80 chosen problems instances from MPSPLib. The problems are used in our empirical experiments in Chapter 5 and 6.

Table B.1: Properties of the chosen 80 problem instances from MPSPLib

Alias	Problem instance	No. of real activities per project	No. of projects	No. of different instances	No. of release times	No. of resource types
I90/2/1	mp-j90_a2_nr5_AC1	90	2	2	2	4
I90/2/2	mp-j90_a2_nr5_AC2	90	2	2	2	4
I90/2/3	mp-j90_a2_nr5_AC3	90	2	2	2	4
I90/2/4	mp-j90_a2_nr5_AC4	90	2	1	2	4
I90/2/5	mp-j90_a2_nr5_AC5	90	2	1	2	4
I90/2/6	mp-j90_a2_nr5_AC6	90	2	2	1	4
I90/2/7	mp-j90_a2_nr5_AC7	90	2	2	1	4
I90/2/8	mp-j90_a2_nr5_AC8	90	2	2	1	4
I90/2/9	mp-j90_a2_nr5_AC9	90	2	1	1	4
I90/2/10	mp-j90_a2_nr5_AC10	90	2	1	1	4
I90/5/1	mp-j90_a5_nr5_AC1	90	5	5	5	4
I90/5/2	mp-j90_a5_nr5_AC2	90	5	5	5	4
I90/5/3	mp-j90_a5_nr5_AC3	90	5	5	5	4
I90/5/4	mp-j90_a5_nr5_AC4	90	5	1	5	4
I90/5/5	mp-j90_a5_nr5_AC5	90	5	1	5	4
I90/5/6	mp-j90_a5_nr5_AC6	90	5	5	1	4
I90/5/7	mp-j90_a5_nr5_AC7	90	5	5	1	4
I90/5/8	mp-j90_a5_nr5_AC8	90	5	5	1	4
I90/5/9	mp-j90_a5_nr5_AC9	90	5	1	1	4
I90/5/10	mp-j90_a5_nr5_AC10	90	5	1	1	4

Table B.1 – continued from previous page

Alias	Problem instance	No. of real activities per project	No. of projects	No. of different instances	No. of release times	No. of resource types
I90/10/1	mp-j90.a10.nr5_AC1	90	10	10	10	4
I90/10/2	mp-j90.a10.nr5_AC2	90	10	10	10	4
I90/10/3	mp-j90.a10.nr5_AC3	90	10	10	10	4
I90/10/4	mp-j90.a10.nr5_AC4	90	10	1	10	4
I90/10/5	mp-j90.a10.nr5_AC5	90	10	1	10	4
I90/10/6	mp-j90.a10.nr5_AC6	90	10	10	1	4
I90/10/7	mp-j90.a10.nr5_AC7	90	10	10	1	4
I90/10/8	mp-j90.a10.nr5_AC8	90	10	10	1	4
I90/10/9	mp-j90.a10.nr5_AC9	90	10	1	1	4
I90/10/10	mp-j90.a10.nr5_AC10	90	10	1	1	4
I90/20/1	mp-j90.a20.nr5_AC1	90	20	10	10	4
I90/20/2	mp-j90.a20.nr5_AC2	90	20	10	10	4
I90/20/3	mp-j90.a20.nr5_AC3	90	20	10	10	4
I90/20/4	mp-j90.a20.nr5_AC4	90	20	1	10	4
I90/20/5	mp-j90.a20.nr5_AC5	90	20	1	10	4
I90/20/6	mp-j90.a20.nr5_AC6	90	20	10	1	4
I90/20/7	mp-j90.a20.nr5_AC7	90	20	10	1	4
I90/20/8	mp-j90.a20.nr5_AC8	90	20	10	1	4
I90/20/9	mp-j90.a20.nr5_AC9	90	20	1	1	4
I90/20/10	mp-j90.a20.nr5_AC10	90	20	1	1	4
I120/2/1	mp-j120.a2.nr5_AC1	120	2	2	2	4
I120/2/2	mp-j120.a2.nr5_AC2	120	2	2	2	4
I120/2/3	mp-j120.a2.nr5_AC3	120	2	2	2	4
I120/2/4	mp-j120.a2.nr5_AC4	120	2	2	2	4
I120/2/5	mp-j120.a2.nr5_AC5	120	2	2	2	4
I120/2/6	mp-j120.a2.nr5_AC6	120	2	2	1	4
I120/2/7	mp-j120.a2.nr5_AC7	120	2	2	1	4
I120/2/8	mp-j120.a2.nr5_AC8	120	2	2	1	4
I120/2/9	mp-j120.a2.nr5_AC9	120	2	2	1	4
I120/2/10	mp-j120.a2.nr5_AC10	120	2	2	1	4
I120/5/1	mp-j120.a5.nr5_AC1	120	5	5	5	4
I120/5/2	mp-j120.a5.nr5_AC2	120	5	5	5	4
I120/5/3	mp-j120.a5.nr5_AC3	120	5	5	5	4
I120/5/4	mp-j120.a5.nr5_AC4	120	5	5	5	4
I120/5/5	mp-j120.a5.nr5_AC5	120	5	5	5	4
I120/5/6	mp-j120.a5.nr5_AC6	120	5	5	1	4
I120/5/7	mp-j120.a5.nr5_AC7	120	5	5	1	4
I120/5/8	mp-j120.a5.nr5_AC8	120	5	5	1	4
I120/5/9	mp-j120.a5.nr5_AC9	120	5	5	1	4
I120/5/10	mp-j120.a5.nr5_AC10	120	5	5	1	4
I120/10/1	mp-j120.a10.nr5_AC1	120	10	10	10	4
I120/10/2	mp-j120.a10.nr5_AC2	120	10	10	5	4

Table B.1 – continued from previous page

Alias	Problem instance	No. of real activities per project	No. of projects	No. of different instances	No. of release times	No. of resource types
I120/10/3	mp_j120_a10_nr5_AC3	120	10	10	5	4
I120/10/4	mp_j120_a10_nr5_AC4	120	10	10	5	4
I120/10/5	mp_j120_a10_nr5_AC5	120	10	10	5	4
I120/10/6	mp_j120_a10_nr5_AC6	120	10	10	1	4
I120/10/7	mp_j120_a10_nr5_AC7	120	10	10	1	4
I120/10/8	mp_j120_a10_nr5_AC8	120	10	10	1	4
I120/10/9	mp_j120_a10_nr5_AC9	120	10	10	1	4
I120/10/10	mp_j120_a10_nr5_AC10	120	10	10	2	4
I120/20/1	mp_j120_a20_nr5_AC1	120	20	20	6	4
I120/20/2	mp_j120_a20_nr5_AC2	120	20	20	10	4
I120/20/3	mp_j120_a20_nr5_AC3	120	20	20	10	4
I120/20/4	mp_j120_a20_nr5_AC4	120	20	20	10	4
I120/20/5	mp_j120_a20_nr5_AC5	120	20	20	10	4
I120/20/6	mp_j120_a20_nr5_AC6	120	20	20	1	4
I120/20/7	mp_j120_a20_nr5_AC7	120	20	20	1	4
I120/20/8	mp_j120_a20_nr5_AC8	120	20	20	1	4
I120/20/9	mp_j120_a20_nr5_AC9	120	20	20	1	4
I120/20/10	mp_j120_a20_nr5_AC10	120	20	20	1	4

Summary

In the preface we stated that the classical decision theory in project management with a single decision maker soon becomes inapplicable because of the large-scale informational and managerial decentralisation. This prediction is thoroughly investigated in the thesis. Obviously, the rapid change in both technology and the structure of the market place in recent years has called for new paradigms for managing large and distributed projects. Within the field of distributed artificial intelligence, the research area of multiagent systems provide a natural way to model and solve problems with inherent complexity that is caused by large-scale decentralisation.

Our research starts from a practical problem of such a decentralised setting — scheduling airport ground handling (AGH) operations. At an airport, many aircraft are turning around at the same time. Each of the aircraft turnaround processes can be seen as a project involving a multitude of organisations working simultaneously on diverse activities. The general goal of our research is to investigate the characteristics of the AGH scheduling problem and provide an adequate solution model that can solve the problem efficiently and robustly. Our proposed multiagent scheduling system, that is discussed in this thesis, may be used to solve a wider range of real-world scheduling problems.

In this thesis, we aim to investigate approaches within a multiagent-system solution framework for designing such a scheduling system. Since agents have only a limited view and control of the overall scheduling problem, Chapter 1 presents the following problem statement.

PS: *Can a number of self-interested agents, by coordinating their local scheduling decisions, achieve a global AGH schedule that is both efficient and robust?*

To answer the problem statement we formulate in Chapter 1 three research questions. They deal with (1) agent-based modelling for AGH scheduling problems, (2) schedule efficiency and robustness under partial observability, and (3) schedule efficiency and robustness under nondeterminism. Precise formulations of the three research questions are given later in this summary. In the remainder of Chapter 1, a five-step research methodology is presented.

In Chapter 2, we identify the characteristics of an AGH scheduling problem and reformulate the problem into a more generic problem, viz. that of a project scheduling problem. A formal description is presented and a range of extensions and variations is discussed. We

reformulate the AGH scheduling problem as a decentralised resource-constrained multi-project scheduling problem under uncertainty (DRCMPSP/u), in which uncertainty is categorised into two classes: (1) partial observability, and (2) nondeterminism.

Chapter 3 reviews the existing solution methods in the literature of project scheduling problems in both OR and AI research. We focus on presenting the state-of-the-art solution methods for solving (1) multi-project scheduling problems, (2) decentralised scheduling problems, and (3) project scheduling under uncertainty. We discuss the limitations of the reviewed solution methods and their (in)applicabilities for solving the AGH scheduling problem. The discussion leads us to a new agent-based model, which we call a lease-based multiagent model.

In Chapter 4, we start to address the first research question, which reads as follows.

RQ1: *How can an AGH scheduling problem be represented in an agent-based model?*

To answer this research question, we propose a novel agent-based model that adopts a ‘coarse-grained’ physical-entity-oriented modelling approach to represent the agents. The model consists of the roles, schedules, and utilities of two classes of agents — resource agents and project agents. We design a market-based coordination mechanism in which the scheduling decisions of the individual agents are coordinated in a lease-based negotiation scenario. The proposed agent representation and inter-agent coordination mechanism offer properties such as self-interestedness, flexibility, and scalability to the agent-based scheduling system.

Chapter 5 addresses the second research question, which reads as follows.

RQ2: *How can agents make and coordinate their local decisions in order to achieve a globally efficient and robust schedule in a partially observable environment?*

To answer this research question, we propose an online iterative scheduling approach in the multiagent setting, called OI-MAS. This approach is composed of (1) a clairvoyant online schedule-generation scheme (COSGS), and (2) an iterative schedule-improvement method (ISIM). By employing the approach and experimenting with it in 80 benchmark problems and 10 simulated AGH scheduling problems, we demonstrate the efficiency and robustness of the resulting schedules.

In Chapter 6, we address the third research question, which reads as follows.

RQ3: *How can agents make and coordinate their local decisions in order to achieve a globally efficient and robust schedule in a nondeterministic environment?*

To answer this research question, we investigate proactive scheduling procedures for constructing stable baseline schedules. In the proactive procedure, different approaches (heuristics and evolutionary learning approaches) are proposed for the two classes of agents to construct stable baseline schedules. A scheduling environment is simulated where the processing times of activities are nondeterministic. By conducting experiments

with the proposed approaches, we show that the constructed schedules are both efficient and robust.

The last chapter of the thesis contains the research conclusions and recommendations for future research. We show that (1) the proposed agent-based model for AGH scheduling problems has the desired system properties such as self-interestedness, flexibility, and scalability; (2) the proposed OI-MAS approach enables the agents to construct efficient and robust schedules under partial observability; and (3) the proposed heuristics and learning approaches enable the agents to construct efficient and robust schedules under nondeterminism. Taking the conclusions as answers to the three corresponding research questions, we are able to give an *affirmative* answer to our problem statement. Lastly, we provide a short discussion on three potential future research lines.

Samenvatting

Klassieke beslissingstheorieën die uitgaan van een centrale beslisser zijn in de praktijk niet direct toepasbaar op het operationele projectmanagement van productie- en dienstverleningprocessen. Dit komt door dat de huidige bedrijfs-informatiesystemen en organisaties heterogeen en gedistribueerd zijn. Bovendien worden hun interoperabiliteit en samenwerking beperkt doordat bepaalde informatie of kennis niet gedeeld wordt (bijv. vanwege competitie) en externe factoren die van invloed zijn op de processen vaak onvoorspelbaar zijn, zoals het weer. Voorts eisen de snelle technologische en markt-ontwikkelingen op het gebied van productie- en dienstverleningsprocessen nieuwe flexibele, efficiënte en robuuste oplossingen voor grootschalig en gedistribueerd projectmanagement. Eerdere studies aangaande multiagentsystemen hebben veelbelovende resultaten opgeleverd en doen vermoeden dat deze systemen een goed alternatief zijn voor centrale beslissers om complexe projectmanagement problemen te modelleren en op te lossen.

Deze dissertatie onderzoekt het vermoeden diepgand. Het gaat daarbij uit van een praktisch gedecentraliseerd planningsprobleem: het plannen van grondafhandeling op een vliegveld (Airport Ground Handling - AGH). Er wordt onderzocht of het mogelijk is een planningssysteem te ontwerpen op basis van de bestaande theorie over multiagentsystemen dat leidt tot een efficiënt en robuust AGH-projectmanagement.

Hiertoe wordt in hoofdstuk 1 het toepassingsdomein geïntroduceerd. Op een vliegveld wordt een groot aantal vliegtuigen tegelijkertijd afgehandeld wat betreft schoonmaken, bevoorrading, bijtanken, veiligheidscontroles, etc. Ieder vliegtuig kan worden gezien als een individueel project, waarbinnen meerdere organisaties tegelijkertijd verschillende activiteiten uitvoeren.

Dit leidt tot de volgende probleemstelling.

PS: *Kan een verzameling agenten, handelend vanuit eigenbelang, een efficiënte en robuuste globale planning voor grondafhandeling maken, door slechts hun eigen lokale planningsbeslissingen te coördineren?*

Om antwoord te geven op de probleemstelling, worden drie onderzoeksvragen geformuleerd. Zij zijn gericht op (1) multiagent modellering van AGH-planningsproblemen, (2) efficiëntie en robuustheid van de planning in geval van gedeeltelijk inzicht, en (3) efficiëntie en robuustheid van de planning onder onvoorspelbare factoren. Om tot een verantwoord onderzoek te komen wordt aan het eind de onderzoeksmethodologie gepresenteerd.

In hoofdstuk 2 worden de kenmerken van een AGH-planningsprobleem geïdentificeerd. Het probleem wordt geëxtrapoleerd naar een generieke problematiek: het projectplan-

ningsprobleem. De formele beschrijving van het projectplanningsprobleem wordt gegeven en enkele variaties en uitbreidingen besproken. Het AGH-planningsprobleem wordt opnieuw geformuleerd als een gedecentraliseerd multi-projectplanningsprobleem met beperkte middelen en onzekerheid (*decentralised resource-Constrained multi-project scheduling problem under uncertainty*). Hierbij wordt onzekerheid in twee klassen ingedeeld: (1) gedeeltelijk inzicht en (2) onvoorspelbaarheid.

Hoofdstuk 3 bespreekt bestaande OR- en AI- oplossingsmethoden uit de literatuur voor projectplanningsproblemen. We presenteren *state-of-the-art* methoden voor het oplossen van (1) multi-project planningsproblemen, (2) gedecentraliseerde planningsproblemen, en (3) projectplanning met onzekerheid. We bespreken de beperkingen van de beschouwde methoden en hun (on)toepasbaarheid voor het oplossen van het AGH-planningsprobleem. Uit deze bespreking wordt een nieuw multiagent model gedis tilleerd, het *lease-based multiagent model*.

In hoofdstuk 4 wordt de eerste onderzoeksvraag besproken.

OV1: *Hoe kan een AGH-planningsprobleem worden gerepresenteerd in een multiagentmodel?*

Ter beantwoording van deze onderzoeksvraag wordt een multiagentmodel voorgesteld, waarin de agenten worden gerepresenteerd via een *physical-entity-oriented* modellering met lage granulariteit. Het model bevat de rollen, planningen en *utilities* van twee klassen van agenten: *resource agents* en *project agents*. Een op de markt gebaseerd mechanisme wordt ontworpen, waarin de planningsbeslissingen van de individuele agenten worden gecoördineerd in een op *lease* gebaseerd onderhandelingscenario. Het voorgestelde mechanisme voor representatie en coördinatie van agenten levert een multiagentplanningssysteem op met als eigenschappen *self-interestedness*, flexibiliteit en schaalbaarheid.

Hoofdstuk 5 behandelt de tweede onderzoeksvraag.

OV2: *Hoe kunnen agenten lokaal beslissingen maken en coördineren, om een globaal efficiënte en robuuste planning te maken, wanneer zij slechts inzicht hebben in een deel van het probleem?*

In antwoord op deze onderzoeksvraag wordt een online, iteratieve planningsaanpak in een multiagent omgeving (d.i., OI-MAS) voorgesteld. Deze aanpak bestaat uit (1) een alleswetend online planninggeneratie-schema (Clairvoyant Online Schedule-Generation Scheme (COSGS)) en (2) een iteratief planningverbeteringsmethode (Iterative Schedule Improvement Method (ISIM)). Deze aanpak wordt toegepast op 80 benchmark problemen en 10 gesimuleerde AGH-planningsproblemen. Zo wordt de efficiëntie en robuustheid van de gemaakte planningen aangetoond.

In hoofdstuk 6 wordt de derde onderzoeksvraag behandeld.

OV3: *Hoe kunnen agenten lokaal beslissingen maken en coördineren, om een globaal efficiënte en robuuste planning te maken, onder onvoorspelbare systeemcondities?*

Pro-actieve planningsprocedures worden ingezet om stabiele *baseline*-planningen te construeren. Hiertoe worden verschillende aanpakken voorgesteld (heuristieken en evolutionair leren). Een AGH-planningsprobleem wordt gesimuleerd onder onvoorspelbare

verwerkingstijden van activiteiten. Door middel van experimenten wordt aangetoond dat de geconstrueerde planningen efficiënt en robuust zijn.

In het laatste hoofdstuk worden conclusies getrokken, en aanbevelingen voor verder onderzoek gedaan. Onze drie conclusies zijn: (1) de voorgestelde multiagent modellen voor AGH-planningsproblemen hebben de gewenste eigenschappen, zoals *self-interestedness*, flexibiliteit en schaalbaarheid; (2) de voorgestelde OI-MAS aanpak stelt de agenten in staat om, zoals met een beperkt inzicht, efficiënte en robuuste planningen te produceren; en (3) de voorgestelde heuristische aanpak in combinatie met de gekozen leeraanpak stelt de agenten in staat efficiënte en robuuste planningen te maken in geval van onvoorspelbaarheid. De conclusies zijn tevens de antwoorden op de drie onderzoeksvragen. Ze leiden naar een *bevestigend* antwoord op de centrale probleemstelling. Tot slot worden nog drie potentiële verdere onderzoeksrichtingen voorgesteld.

Curriculum Vitae

Xiaoyu Mao was born in Xi'an, Shaanxi, P.R. China, on December 6, 1980. In the same city, he attended primary school and high school (Xi'an Tie Yi High School, graduated in 1999). He then started his Bachelor study on Computer Science and Technology at the School of Computer Science in Northwestern Polytechnical University, China. After obtaining his Bachelor degree with honours in July 2003, he travelled to France and started a six-month transition semester on French-language learning and Electronic Engineering study at Ecole Spéciale de Mécanique et d'Electricité (ESME-Sudria), Paris. This study was followed by an Engineering Diploma pursuing on Real-time and Systems at Institut National des Sciences Appliquées, Toulouse (INSA-Toulouse). In addition to the engineering studies, he enrolled himself in a Research Master program on Critical Information System and Network. After his graduation with double diplomas (i.e., a French Engineering Diploma and a Master of Science) in 2005, he accepted a position as a Ph.D. researcher in Casimir program at Almende B.V., a private innovative research company located in Rotterdam, the Netherlands. His research was co-supervised by Professor Jaap van den Herik and Professor Eric Postma at Tilburg Center for Cognition and Communication (TiCC), Tilburg University, as well as by Doctor Nico Roos at the Department of Knowledge Engineering (DKE), Maastricht University and Doctor Alfons Salden at Almende B.V. Currently, Xiaoyu works as a technology consultant at ASK Community Systems B.V., Rotterdam.

List of Publications

The investigations performed during this PhD research resulted in the following publications.

Mao, X., ter Mors, A., Roos, N., and Witteveen, C. (2006). Agent-based scheduling for aircraft deicing. In P.-Y. Schobbens, W. Vanhoof, and G. Schwanen (Eds.) *Proceedings of the 18th Belgium—Netherlands Conference on Artificial Intelligence*, (pp. 229-236). BNVKI.

ter Mors, A., Mao, X., Roos, N., Witteveen, C., and Salden, A. (2007a). Multi-agent system support for scheduling aircraft de-icing. In *Proceeding of ISCRAM 2007 - Intelligent Human Computer Systems for Crisis Response and Management*, (pp. 467-478).

Mao, X., ter Mors, A., Roos, N., and Witteveen, C. (2007b). Coordinating competitive agents in dynamic airport resource scheduling. In P. Petta, J. P. Müller, M. Klusch, and M. P. Georgeff (Eds.) *Multiagent System Technologies, 5th German Conference, MATES 2007, Leipzig, Germany, September 24-26, 2007, Proceedings*, (pp. 133-144).

Mao, X., ter Mors, A., Roos, N., and Witteveen, C. (2007c). Using neuro-evolution in aircraft deicing scheduling. In K. Tuyls, S. de Jong, M. Ponsen, and K. Verbeeck (Eds.) *ALAMAS 2007 Adaptive and Learning Agents and Multi-Agent Systems*, (pp. 138-145).

ter Mors, A. W., Mao, X., Zutt, J., Witteveen, C., and Roos, N. (2008a). Robust reservation-based multi-agent routing. In M. Ghallab, C. Spyropoulos, N. Fakotakis, and N. Avouris (Eds.) *Proceedings of the 18th European Conference on Artificial Intelligence*, (pp. 929-930). IOS Press, Amsterdam. Accepted as poster.

Mao, X., Roos, N., and Salden, A. (2008b). Distribute the Selfish Ambitions. In A. Nijholt, M. Pantic, M. Poel, and H. Hondorp (Eds.) *Proceedings of the 20th Belgian—Dutch Conference on Artificial Intelligence*, (pp. 137-144). BNVKI.

Mao, X., Roos, N., and Salden, A. (2009). Stable Multi-project Scheduling of Airport Ground Handling Services with Heterogeneous Agents, In Decher, Sichman, Sierra, and Castelfranchi (Eds.) *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS09)*, May, 10-15, 2009, Budapest, Hungary, (pp. 537-544).

SIKS Dissertation Series

1998

- 1 Johan van den Akker (CWI¹) *DEGAS - An Active, Temporal Database of Autonomous Objects*
- 2 Floris Wiesman (UM) *Information Retrieval by Graphically Browsing Meta-Information*
- 3 Ans Steuten (TUD) *A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective*
- 4 Dennis Breuker (UM) *Memory versus Search in Games*
- 5 Eduard W. Oskamp (RUL) *Computerondersteuning bij Straftoemeting*

1999

- 1 Mark Sloof (VU) *Physiology of Quality Change Modelling; Automated Modelling of Quality Change of Agricultural Products*
- 2 Rob Potharst (EUR) *Classification using Decision Trees and Neural Nets*
- 3 Don Beal (UM) *The Nature of Minimax Search*
- 4 Jacques Penders (UM) *The Practical Art of Moving Physical Objects*
- 5 Aldo de Moor (KUB) *Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems*
- 6 Niek J.E. Wijngaards (VU) *Re-Design of Compositional Systems*
- 7 David Spelt (UT) *Verification Support for Object Database Design*

- 8 Jacques H.J. Lenting (UM) *Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation*

2000

- 1 Frank Niessink (VU) *Perspectives on Improving Software Maintenance*
- 2 Koen Holtman (TU/e) *Prototyping of CMS Storage Management*
- 3 Carolien M.T. Metselaar (UvA) *Sociaal-organisatorische Gevolgen van Kennistechnologie; een Procesbenadering en Actorperspectief*
- 4 Geert de Haan (VU) *ETAG, A Formal Model of Competence Knowledge for User Interface Design*
- 5 Ruud van der Pol (UM) *Knowledge-Based Query Formulation in Information Retrieval*
- 6 Rogier van Eijk (UU) *Programming Languages for Agent Communication*
- 7 Niels Peek (UU) *Decision-Theoretic Planning of Clinical Patient Management*
- 8 Veerle Coupé (EUR) *Sensitivity Analysis of Decision-Theoretic Networks*
- 9 Florian Waas (CWI) *Principles of Probabilistic Query Optimization*
- 10 Niels Nes (CWI) *Image Database Management System Design Considerations, Algorithms and Architecture*
- 11 Jonas Karlsson (CWI) *Scalable Distributed Data Structures for Database Management*

¹Abbreviations: SIKS - Dutch Research School for Information and Knowledge Systems; CWI - Centrum voor Wiskunde en Informatica, Amsterdam; EUR - Erasmus Universiteit, Rotterdam; KUB - Katholieke Universiteit Brabant, Tilburg; KUN - Katholieke Universiteit Nijmegen; OU - Open Universiteit; RUL - Rijksuniversiteit Leiden; RUN - Radboud Universiteit Nijmegen; TUD - Technische Universiteit Delft; TU/e - Technische Universiteit Eindhoven; UL - Universiteit Leiden; UM - Universiteit Maastricht; UT - Universiteit Twente, Enschede; UU - Universiteit Utrecht; UvA - Universiteit van Amsterdam; UvT - Universiteit van Tilburg; VU - Vrije Universiteit, Amsterdam.

2001

- 1 Silja Renooij (UU) *Qualitative Approaches to Quantifying Probabilistic Networks*
- 2 Koen Hindriks (UU) *Agent Programming Languages: Programming with Mental Models*
- 3 Maarten van Someren (UvA) *Learning as Problem Solving*
- 4 Evgueni Smirnov (UM) *Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets*
- 5 Jacco van Ossensbruggen (VU) *Processing Structured Hypermedia: A Matter of Style*
- 6 Martijn van Welie (VU) *Task-Based User Interface Design*
- 7 Bastiaan Schonhage (VU) *Diva: Architectural Perspectives on Information Visualization*
- 8 Pascal van Eck (VU) *A Compositional Semantic Structure for Multi-Agent Systems Dynamics*
- 9 Pieter Jan 't Hoen (RUL) *Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes*
- 10 Maarten Sierhuis (UvA) *Modeling and Simulating Work Practice BRAHMS: a Multiagent Modeling and Simulation Language for Work Practice Analysis and Design*
- 11 Tom M. van Engers (VU) *Knowledge Management: The Role of Mental Models in Business Systems Design*

2002

- 1 Nico Lassing (VU) *Architecture-Level Modifiability Analysis*
- 2 Roelof van Zwol (UT) *Modelling and Searching Web-based Document Collections*
- 3 Henk Ernst Blok (UT) *Database Optimization Aspects for Information Retrieval*
- 4 Juan Roberto Castelo Valdueza (UU) *The Discrete Acyclic Digraph Markov Model in Data Mining*
- 5 Radu Serban (VU) *The Private Cyberspace Modeling Electronic Environments Inhabited by Privacy-Concerned Agents*
- 6 Laurens Mommers (UL) *Applied Legal Epistemology; Building a Knowledge-based Ontology of the Legal Domain*
- 7 Peter Boncz (CWI) *Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications*
- 8 Jaap Gordijn (VU) *Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas*

- 9 Willem-Jan van den Heuvel (KUB) *Integrating Modern Business Applications with Objectified Legacy Systems*
- 10 Brian Sheppard (UM) *Towards Perfect Play of Scrabble*
- 11 Wouter C.A. Wijngaards (VU) *Agent Based Modelling of Dynamics: Biological and Organizational Applications*
- 12 Albrecht Schmidt (UvA) *Processing XML in Database Systems*
- 13 Hongjing Wu (TU/e) *A Reference Architecture for Adaptive Hypermedia Applications*
- 14 Wieke de Vries (UU) *Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems*
- 15 Rik Eshuis (UT) *Semantics and Verification of UML Activity Diagrams for Workflow Modelling*
- 16 Pieter van Langen (VU) *The Anatomy of Design: Foundations, Models and Applications*
- 17 Stefan Manegold (UvA) *Understanding, Modeling, and Improving Main-Memory Database Performance*

2003

- 1 Heiner Stuckenschmidt (VU) *Ontology-Based Information Sharing in Weakly Structured Environments*
- 2 Jan Broersen (VU) *Modal Action Logics for Reasoning About Reactive Systems*
- 3 Martijn Schuemie (TUD) *Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy*
- 4 Milan Petkovic (UT) *Content-Based Video Retrieval Supported by Database Technology*
- 5 Jos Lehmann (UvA) *Causation in Artificial Intelligence and Law – A Modelling Approach*
- 6 Boris van Schooten (UT) *Development and Specification of Virtual Environments*
- 7 Machiel Jansen (UvA) *Formal Explorations of Knowledge Intensive Tasks*
- 8 Yong-Ping Ran (UM) *Repair-Based Scheduling*
- 9 Rens Kortmann (UM) *The Resolution of Visually Guided Behaviour*
- 10 Andreas Lincke (UT) *Electronic Business Negotiation: Some Experimental Studies on the Interaction between Medium, Innovation Context and Cult*
- 11 Simon Keizer (UT) *Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks*

- 12 Roeland Ordelman (UT) *Dutch Speech Recognition in Multimedia Information Retrieval*
- 13 Jeroen Donkers (UM) *Nosce Hostem – Searching with Opponent Models*
- 14 Stijn Hoppenbrouwers (KUN) *Freezing Language: Conceptualisation Processes across ICT-Supported Organisations*
- 15 Mathijs de Weerdt (TUD) *Plan Merging in Multi-Agent Systems*
- 16 Menzo Windhouwer (CWI) *Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouse*
- 17 David Jansen (UT) *Extensions of Statecharts with Probability, Time, and Stochastic Timing*
- 18 Levente Kocsis (UM) *Learning Search Decisions*
- 14 Paul Harrenstein (UU) *Logic in Conflict. Logical Explorations in Strategic Equilibrium*
- 15 Arno Knobbe (UU) *Multi-Relational Data Mining*
- 16 Federico Divina (VU) *Hybrid Genetic Relational Search for Inductive Learning*
- 17 Mark Winands (UM) *Informed Search in Complex Games*
- 18 Vania Bessa Machado (UvA) *Supporting the Construction of Qualitative Knowledge Models*
- 19 Thijs Westerveld (UT) *Using generative probabilistic models for multimedia retrieval*
- 20 Madelon Evers (Nyenrode) *Learning from Design: facilitating multidisciplinary design teams*

2004

- 1 Virginia Dignum (UU) *A Model for Organizational Interaction: Based on Agents, Founded in Logic*
- 2 Lai Xu (UvT) *Monitoring Multi-party Contracts for E-business*
- 3 Perry Groot (VU) *A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving*
- 4 Chris van Aart (UvA) *Organizational Principles for Multi-Agent Architectures*
- 5 Viara Popova (EUR) *Knowledge Discovery and Monotonicity*
- 6 Bart-Jan Hommes (TUD) *The Evaluation of Business Process Modeling Techniques*
- 7 Elise Boltjes (UM) *Voorbeeld_{IG} Onderwijs; Voorbeeldgestuurd Onderwijs, een Opstap naar Abstract Denken, vooral voor Meisjes*
- 8 Joop Verbeek (UM) *Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale Politieke Gegevensuitwisseling en Digitale Expertise*
- 9 Martin Caminada (VU) *For the Sake of the Argument; Explorations into Argument-based Reasoning*
- 10 Suzanne Kabel (UvA) *Knowledge-rich Indexing of Learning-objects*
- 11 Michel Klein (VU) *Change Management for Distributed Ontologies*
- 12 The Duy Bui (UT) *Creating Emotions and Facial Expressions for Embodied Agents*
- 13 Wojciech Jamroga (UT) *Using Multiple Models of Reality: On Agents who Know how to Play*
- 1 Floor Verdenius (UvA) *Methodological Aspects of Designing Induction-Based Applications*
- 2 Erik van der Werf (UM) *AI techniques for the game of Go*
- 3 Franc Grootjen (RUN) *A Pragmatic Approach to the Conceptualisation of Language*
- 4 Nirvana Meratnia (UT) *Towards Database Support for Moving Object data*
- 5 Gabriel Infante-Lopez (UvA) *Two-Level Probabilistic Grammars for Natural Language Parsing*
- 6 Pieter Spronck (UM) *Adaptive Game AI*
- 7 Flavius Frasinicar (TU/e) *Hypermedia Presentation Generation for Semantic Web Information Systems*
- 8 Richard Vdovjak (TU/e) *A Model-driven Approach for Building Distributed Ontology-based Web Applications*
- 9 Jeen Broekstra (VU) *Storage, Querying and Inferencing for Semantic Web Languages*
- 10 Anders Bouwer (UvA) *Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments*
- 11 Elth Ogston (VU) *Agent Based Matchmaking and Clustering - A Decentralized Approach to Search*
- 12 Csaba Boer (EUR) *Distributed Simulation in Industry*
- 13 Fred Hamburg (UL) *Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen*
- 14 Borys Omelayenko (VU) *Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics*
- 15 Tibor Bosse (VU) *Analysis of the Dynamics of Cognitive Processes*

- 16 Joris Graaumanns (UU) *Usability of XML Query Languages*
- 17 Boris Shishkov (TUD) *Software Specification Based on Re-usable Business Components*
- 18 Danielle Sent (UU) *Test-selection strategies for probabilistic networks*
- 19 Michel van Dartel (UM) *Situated Representation*
- 20 Cristina Coteanu (UL) *Cyber Consumer Law, State of the Art and Perspectives*
- 21 Wijnand Derks (UT) *Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics*
- 16 Carsten Riggelsen (UU) *Approximation Methods for Efficient Learning of Bayesian Networks*
- 17 Stacey Nagata (UU) *User Assistance for Multi-tasking with Interruptions on a Mobile Device*
- 18 Valentin Zhizhikun (UvA) *Graph transformation for Natural Language Processing*
- 19 Birna van Riemsdijk (UU) *Cognitive Agent Programming: A Semantic Approach*
- 20 Marina Velikova (UvT) *Monotone models for prediction in data mining*
- 21 Bas van Gils (RUN) *Aptness on the Web*
- 22 Paul de Vrieze (RUN) *Fundamentals of Adaptive Personalisation*

2006

- 1 Samuil Angelov (TU/e) *Foundations of B2B Electronic Contracting*
- 2 Cristina Chisalita (VU) *Contextual issues in the design and use of information technology in organizations*
- 3 Noor Christoph (UvA) *The role of metacognitive skills in learning to solve problems*
- 4 Marta Sabou (VU) *Building Web Service Ontologies*
- 5 Cees Pierik (UU) *Validation Techniques for Object-Oriented Proof Outlines*
- 6 Ziv Baida (VU) *Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling*
- 7 Marko Smiljanic (UT) *XML schema matching – balancing efficiency and effectiveness by means of clustering*
- 8 Eelco Herder (UT) *Forward, Back and Home Again - Analyzing User Behavior on the Web*
- 9 Mohamed Wahdan (UM) *Automatic Formulation of the Auditor's Opinion*
- 10 Ronny Siebes (VU) *Semantic Routing in Peer-to-Peer Systems*
- 11 Joeri van Ruth (UT) *Flattening Queries over Nested Data Types*
- 12 Bert Bongers (VU) *Interactivation - Towards an e-cology of people, our technological environment, and the arts*
- 13 Henk-Jan Lebbink (UU) *Dialogue and Decision Games for Information Exchanging Agents*
- 14 Johan Hoorn (VU) *Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change*
- 15 Rainer Malik (UU) *CONAN: Text Mining in the Biomedical Domain*
- 23 Ion Juvina (UU) *Development of Cognitive Model for Navigating on the Web*
- 24 Laura Hollink (VU) *Semantic Annotation for Retrieval of Visual Resources*
- 25 Madalina Drugan (UU) *Conditional log-likelihood MDL and Evolutionary MCMC*
- 26 Vojkan Mihajlovic (UT) *Score Region Algebra: A Flexible Framework for Structured Information Retrieval*
- 27 Stefano Bocconi (CWI) *Vox Populi: generating video documentaries from semantically annotated media repositories*
- 28 Borkur Sigurbjornsson (UvA) *Focused Information Access using XML Element Retrieval*

2007

- 1 Kees Leune (UvT) *Access Control and Service-Oriented Architectures*
- 2 Wouter Teepe (RUG) *Reconciling Information Exchange and Confidentiality: A Formal Approach*
- 3 Peter Mika (VU) *Social Networks and the Semantic Web*
- 4 Jurriaan van Diggelen (UU) *Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach*
- 5 Bart Schermer (UL) *Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance*
- 6 Gilad Mishne (UvA) *Applied Text Analytics for Blogs*
- 7 Natasa Jovanovic' (UT) *To Whom It May Concern - Addressee Identification in Face-to-Face Meetings*
- 8 Mark Hoogendoorn (VU) *Modeling of Change in Multi-Agent Organizations*

- 9 David Mobach (VU) *Agent-Based Mediated Service Negotiation*
- 10 Huib Aldewereld (UU) *Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols*
- 11 Natalia Stash (TU/e) *Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System*
- 12 Marcel van Gerven (RUN) *Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty*
- 13 Rutger Rienks (UT) *Meetings in Smart Environments; Implications of Progressing Technology*
- 14 Niek Bergboer (UM) *Context-Based Image Analysis*
- 15 Joyca Lacroix (UM) *NIM: a Situated Computational Memory Model*
- 16 Davide Grossi (UU) *Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems*
- 17 Theodore Charitos (UU) *Reasoning with Dynamic Networks in Practice*
- 18 Bart Orriens (UvT) *On the development and management of adaptive business collaborations*
- 19 David Levy (UM) *Intimate relationships with artificial partners*
- 20 Slinger Jansen (UU) *Customer Configuration Updating in a Software Supply Network*
- 21 Karianne Vermaas (UU) *Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005*
- 22 Zlatko Zlatev (UT) *Goal-oriented design of value and process models from patterns*
- 23 Peter Barna (TU/e) *Specification of Application Logic in Web Information Systems*
- 24 Georgina Ramírez Camps (CWI) *Structural Features in XML Retrieval*
- 25 Joost Schalken (VU) *Empirical Investigations in Software Process Improvement*
- 5 Vera Hollink (UvA) *Optimizing hierarchical menus: a usage-based approach*
- 4 Ander de Keijzer (UT) *Management of Uncertain Data - towards unattended integration*
- 5 Bela Mutschler (UT) *Modeling and simulating causal dependencies on process-aware information systems from a cost perspective*
- 6 Arjen Hommersom (RUN) *On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective*
- 7 Peter van Rosmalen (OU) *Supporting the tutor in the design and support of adaptive e-learning*
- 8 Janneke Bolt (UU) *Bayesian Networks: Aspects of Approximate Inference*
- 9 Christof van Nimwegen (UU) *The paradox of the guided user: assistance can be counter-effective*
- 10 Wauter Bosma (UT) *Discourse oriented Summarization*
- 11 Vera Kartseva (VU) *Designing Controls for Network Organizations: a Value-Based Approach*
- 12 Jozsef Farkas (RUN) *A Semiotically oriented Cognitive Model of Knowledge Representation*
- 13 Caterina Carraciolo (UvA) *Topic Driven Access to Scientific Handbooks*
- 14 Arthur van Bunningen (UT) *Context-Aware Querying; Better Answers with Less Effort*
- 15 Martijn van Otterlo (UT) *The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains*
- 16 Henriette van Vugt (VU) *Embodied Agents from a User's Perspective*
- 17 Martin Op't Land (TUD) *Applying Architecture and Ontology to the Splitting and Allying of Enterprises*
- 18 Guido de Croon (UM) *Adaptive Active Vision*
- 19 Henning Rode (UT) *From document to entity retrieval: improving precision and performance of focused text search*
- 20 Rex Arendsen (UvA) *Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met een overheid op de administratieve lasten van bedrijven*

2008

- 1 Katalin Boer-Sorbán (EUR) *Agent-Based Simulation of Financial Markets: A modular, continuous-time approach*
- 2 Alexei Sharpanskykh (VU) *On Computer-Aided Methods for Modeling and Analysis of Organizations*
- 21 Krisztian Balog (UvA) *People search in the enterprise*
- 22 Henk Koning (UU) *Communication of IT-architecture*
- 23 Stefan Visscher (UU) *Bayesian network models for the management of ventilator-associated pneumonia*

- 24 Zharko Aleksovski (VU) *Using background knowledge in ontology matching*
 - 25 Geert Jonker (UU) *Efficient and Equitable exchange in air traffic management plan repair using spender-signed currency*
 - 26 Marijn Huijbregts (UT) *Segmentation, diarization and speech transcription: surprise data unraveled*
 - 27 Hubert Vogten (OU) *Design and implementation strategies for IMS learning design*
 - 28 Ildiko Flesh (RUN) *On the use of independence relations in Bayesian networks*
 - 29 Dennis Reidsma (UT) *Annotations and subjective machines- Of annotators, embodied agents, users, and other humans*
 - 30 Wouter van Atteveldt (VU) *Semantic network analysis: techniques for extracting, representing and querying media content*
 - 31 Loes Braun (UM) *Pro-active medical information retrieval*
 - 32 Trung B. Hui (UT) *Toward affective dialogue management using partially observable markov decision processes*
 - 33 Frank Terpstra (UvA) *Scientific workflow design; theoretical and practical issues*
 - 34 Jeroen de Knijf (UU) *Studies in Frequent Tree Mining*
 - 35 Benjamin Torben-Nielsen (UvT) *Dendritic morphology: function shapes structure*
 - 9 Benjamin Kanagwa (RUN) *Design, Discovery and Construction of Service-oriented Systems*
 - 10 Jan Wielemaker (UvA) *Logic programming for knowledge-intensive interactive applications*
 - 11 Alexander Boer (UvA) *Legal Theory, Sources of Law & the Semantic Web*
 - 12 Peter Massuthe (TU/e, Humboldt-Universität zu Berlin) *Operating Guidelines for Services*
 - 13 Steven de Jong (UM) *Fairness in Multi-Agent Systems*
 - 14 Maksym Korotkiy (VU) *From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)*
 - 15 Rinke Hoekstra (UvA) *Ontology Representation - Design Patterns and Ontologies that Make Sense*
 - 16 Fritz Reul (UvT) *New Architectures in Computer Chess*
 - 17 Laurens van der Maaten (UvT) *Feature Extraction from Visual Data*
 - 18 Fabian Groffen (CWI) *Armada, An Evolving Database System*
 - 19 Valentin Robu (CWI) *Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets*
 - 20 Bob van der Vecht (UU) *Adjustable Autonomy: Controlling Influences on Decision Making*
 - 21 Stijn Vanderlooy (UM) *Ranking and Reliable Classification*
 - 22 Pavel Serdyukov (UT) *Search For Expertise: Going beyond direct evidence*
 - 23 Peter Hofgesang (VU) *Modelling Web Usage in a Changing Environment*
 - 24 Annerieke Heuvelink (VU) *Cognitive Models for Training Simulations*
 - 25 Alex van Ballegooij (CWI) *"RAM: Array Database Management through Relational Mapping"*
 - 26 Fernando Koch (UU) *An Agent-Based Model for the Development of Intelligent Mobile Services*
 - 27 Christian Glahn (OU) *Contextual Support of social Engagement and Reflection on the Web*
 - 28 Sander Evers (UT) *Sensor Data Management with Probabilistic Models*
 - 29 Stanislav Pokraev (UT) *Model-Driven Semantic Integration of Service-Oriented Applications*
 - 30 Marcin Zukowski (CWI) *Balancing vectorized query execution with bandwidth-optimized storage*
- 2009**
- 1 Rasa Jurgelenaite (RUN) *Symmetric Causal Independence Models*
 - 2 Willem Robert van Hage (VU) *Evaluating Ontology-Alignment Techniques*
 - 3 Hans Stol (UvT) *A Framework for Evidence-based Policy Making Using IT*
 - 4 Josephine Nabukenya (RUN) *Improving the Quality of Organisational Policy Making using Collaboration Engineering*
 - 5 Sietse Overbeek (RUN) *Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality*
 - 6 Muhammad Subianto (UU) *Understanding Classification*
 - 7 Ronald Poppe (UT) *Discriminative Vision-Based Recovery and Recognition of Human Motion*
 - 8 Volker Nannen (VU) *Evolutionary Agent-Based Policy Analysis in Dynamic Environments*

- 31 Sofiya Katrenko (UvA) *A Closer Look at Learning Relations from Text*
- 32 Rik Farenhorst and Remco de Boer (VU) *Architectural Knowledge Management: Supporting Architects and Auditors*
- 33 Khiet Truong (UT) *How Does Real Affect Affect Affect Recognition In Speech?*
- 34 Inge van de Weerd (UU) *Advancing in Software Product Management: An Incremental Method Engineering Approach*
- 35 Wouter Koelewijn (UL) *Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling*
- 36 Marco Kalz (OU) *Placement Support for Learners in Learning Networks*
- 37 Hendrik Drachslers (OU) *Navigation Support for Learners in Informal Learning Networks*
- 38 Riina Vuorikari (OU) *Tags and self-organisation: a metadata ecology for learning resources in a multilingual context*
- 39 Christian Stahl (TU/e, Humboldt-Universitaet zu Berlin) *Service Substitution — A Behavioral Approach Based on Petri Nets*
- 40 Stephan Raaijmakers (UvT) *Multinomial Language Learning: Investigations into the Geometry of Language*
- 41 Igor Berezhnoy (UvT) *Digital Analysis of Paintings*
- 42 Toine Bogers (UvT) *Recommender Systems for Social Bookmarking*
- 43 Virginia Nunes Leal Franqueira (UT) *Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients*
- 44 Roberto Santana Tapia (UT) *Assessing Business-IT Alignment in Networked Organizations*
- 45 Jilles Vreeken (UU) *Making Pattern Mining Useful*
- 46 Loredana Afanasiev (UvA) *Querying XML: Benchmarks and Recursion*
- 5 Claudia Hauff (UT) *Predicting the Effectiveness of Queries and Retrieval Systems*
- 6 Sander Bakkes (UvT) *Rapid Adaptation of Video Game AI*
- 7 Wim Fikkert (UT) *A Gesture interaction at a Distance*
- 8 Krzysztof Siewicz (UL) *Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments*
- 9 Hugo Kielman (UL) *Politiegegevensverwerking en Privacy, Naar een effectieve waarborging*
- 10 Rebecca Ong (UL) *Mobile Communication and Protection of Children*
- 11 Adriaan Ter Mors (TUD) *The world according to MARP: Multi-Agent Route Planning*
- 12 Susan van den Braak (UU) *Sensemaking software for crime analysis*
- 13 Gianluigi Folino (RUN) *High Performance Data Mining using Bio-inspired techniques*
- 14 Sander van Splunter (VU) *Automated Web Service Reconfiguration*
- 15 Lianne Bodestaff (UT) *Managing Dependency Relations in Inter-Organizational Models*
- 16 Sicco Verwer (TUD) *Efficient Identification of Timed Automata, theory and practice*
- 17 Spyros Kotoulas (VU) *Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications*
- 18 Charlotte Gerritsen (VU) *Caught in the Act: Investigating Crime by Agent-Based Simulation*
- 19 Henriette Cramer (UvA) *People's Responses to Autonomous and Adaptive Systems*
- 20 Ivo Swartjes (UT) *Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative*
- 21 Harold van Heerde (UT) *Privacy-aware data management by means of data degradation*
- 22 Michiel Hildebrand (CWI) *End-user Support for Access to Heterogeneous Linked Data*
- 23 Bas Steunebrink (UU) *The Logical Structure of Emotions*
- 24 Dmytro Tykhonov (TUD) *Designing Generic and Efficient Negotiation Strategies*
- 25 Zulfiqar Ali Memon (VU) *Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective*
- 26 Ying Zhang (CWI) *XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines*

2010

- 1 Matthijs van Leeuwen (UU) *Patterns that Matter*
- 2 Ingo Wassink (UT) *Work flows in Life Science*
- 3 Joost Geurts (CWI) *A Document Engineering Model and Processing Framework for Multimedia documents*
- 4 Olga Kulyk (UT) *Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments*

- 27 Marten Voulon (UL) *Automatisch contracteren*
 - 28 Arne Koopman (UU) *Characteristic Relational Patterns*
 - 29 Stratos Idreos (CWI) *Database Cracking: Towards Auto-tuning Database Kernels*
 - 30 Marieke van Erp (UvT) *Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval*
 - 31 Victor de Boer (UvA) *Ontology Enrichment from Heterogeneous Sources on the Web*
 - 32 Marcel Hiel (UvT) *An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems*
 - 33 Robin Aly (UT) *Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval*
 - 34 Teduh Dirgahayu (UT) *Interaction Design in Service Compositions*
 - 35 Dolf Trieschnigg (UT) *Proof of Concept: Concept-based Biomedical Information Retrieval*
 - 36 Jose Janssen (OU) *Paving the Way for Lifelong Learning; Facilitating competence development through a learning path specification*
 - 37 Niels Lohmann (TU/e) *Correctness of services and their composition*
 - 38 Dirk Fahland (TU/e) *From Scenarios to components*
 - 39 Ghazanfar Farooq Siddiqui (VU) *Integrative modeling of emotions in virtual agents*
 - 40 Mark van Assem (VU) *Converting and Integrating Vocabularies for the Semantic Web*
 - 41 Guillaume Chaslot (UM) *Monte-Carlo Tree Search*
 - 42 Sybren de Kinderen (VU) *Needs-driven service bundling in a multi-supplier setting - the computational e3-service approach*
 - 43 Peter van Kranenburg (UU) *A Computational Approach to Content-Based Retrieval of Folk Song Melodies*
 - 44 Pieter Bellekens (TU/e) *An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain*
 - 45 Vasilios Andrikopoulos (UvT) *A theory and model for the evolution of software services*
 - 46 Vincent Pijpers (VU) *e3alignment: Exploring Inter-Organizational Business-ICT Alignment*
 - 47 Chen Li (UT) *Mining Process Model Variants: Challenges, Techniques, Examples*
 - 48 Milan Lovric (EUR) *Behavioral Finance and Agent-Based Artificial Markets*
 - 49 Jahn-Takeshi Saito (UM) *Solving difficult game positions*
 - 50 Bouke Huurnink (UvA) *Search in Audiovisual Broadcast Archives*
 - 51 Alia Khairia Amin (CWI) *Understanding and supporting information seeking tasks in multiple sources*
 - 52 Peter-Paul van Maanen (VU) *Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention*
 - 53 Edgar Meij (UvA) *Combining Concepts and Language Models for Information Access*
- 2011**
- 1 Botond Cseke (RUN) *Variational Algorithms for Bayesian Inference in Latent Gaussian Models*
 - 2 Nick Tinnemeier (UU) *Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language*
 - 3 Jan Martijn van der Werf (TU/e) *Compositional Design and Verification of Component-Based Information Systems*
 - 4 Hado van Hasselt (UU) *Insights in Reinforcement Learning. Formal analysis and empirical evaluation of temporal-difference learning algorithms*
 - 5 Base van der Raadt (VU) *Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.*
 - 6 Yiwen Wang (TU/e) *Semantically-Enhanced Recommendations in Cultural Heritage*
 - 7 Yujia Cao (UT) *Multimodal Information Presentation for High Load Human Computer Interaction*
 - 8 Nieske Vergunst (UU) *BDI-based Generation of Robust Task-Oriented Dialogues*
 - 9 Tim de Jong (OU) *Contextualised Mobile Media for Learning*
 - 10 Bart Bogaert (UvT) *Cloud Content Contention*
 - 11 Dhaval Vyas (UT) *Designing for Awareness: An Experience-focused HCI Perspective*
 - 12 Carmen Bratosin (TU/e) *Grid Architecture for Distributed Process Mining*
 - 13 Xiaoyu Mao (UvT) *Airport under Control. Multiagent Scheduling for Airport Ground Handling*

TiCC Ph.D. Series

- 1 Pashiera Barkhuysen. *Audiovisual prosody in interaction*. Promotores: M.G.J. Swerts, E.J. Krahmer. Tilburg, 3 October 2008
- 2 Ben Torben-Nielsen. *Dendritic morphology: function shapes structure*. Promotores: H.J. van den Herik, E.O. Postma. Copromotor: K.P. Tuyls. Tilburg, 3 December 2008
- 3 Hans Stol. *A framework for evidence-based policy making using IT. A systems approach*. Promotor: H.J. van den Herik. Tilburg, 21 January 2009
- 4 Jeroen Geertzen. *Act recognition and prediction. Explorations in computational dialogue modelling*. Promotor: H.C. Bunt. Copromotor: J.M.B. Terken. Tilburg, 11 February 2009
- 5 Sander Canisius. *Structural prediction for natural language processing: a constraint satisfaction approach*. Promotores: A.P.J. van den Bosch, W.M.P. Daelemans. Tilburg, 13 February 2009
- 6 Fritz M.H. Reul. *New Architectures in Computer Chess*. Promotor: H.J. van den Herik. Copromotor: J.W.H.M. Uiterwijk. Tilburg, 17 June 2009
- 7 Laurens van der Maaten. *Feature Extraction from Visual Data*. Promotores: E.O. Postma, H.J. van den Herik. Copromotor: A.G. Lange. Tilburg, 23 June 2009
- 8 Stephan Raaijmakers. *Multinomial Language Learning: Investigations into the Geometry of Language*. Promotores: W.M.P. Daelemans, A.P.J. van den Bosch. Tilburg, 1 December 2009
- 9 Igor Berezhnoy. *Digital Analysis of Paintings*. Promotores: E.O. Postma, H.J. van den Herik. Tilburg, 7 December 2009
- 10 Toine Bogers. *Recommender Systems for Social Bookmarking*. Promotor: A.P.J. van den Bosch. Tilburg, 8 December 2009
- 11 Sander Bakkes. *Rapid Adaptation of Video Game AI*. Promotor: H.J. van den Herik. Copromotor: P.H.M. Spronck. Tilburg, 3 March 2010
- 12 Maria Mos. *Complex Lexical Items*. Promotor: A.P.J. van den Bosch. Copromotores: A. Vermeer, A. Backus. Tilburg, 12 May 2010
- 13 Marieke van Erp. *Accessing Natural History. Discoveries in data cleaning, structuring, and retrieval*. Promotor: A.P.J. van den Bosch. Tilburg, 30 June 2010
- 14 Edwin Commandeur. *Implicit causality and implicit consequentially in language comprehension*. Promotores: L.G.M. Noordman, W. Vonk. Copromotor: R. Cozijn. Tilburg, 30 June 2010

- 15 Bart Bogaert. *Cloud Content Contention*. Promotores: H.J. van den Herik, E.O. Postma. Tilburg, 30 March 2011
- 16 Xiaoyu Mao. *Airport under Control. Multiagent Scheduling for Airport Ground Handling*. Promotores: H.J. van den Herik, E.O. Postma. Copromotores: N. Roos, A.H. Salden. Tilburg, 25 May 2011