Robust & decentralized project scheduling

Martens, Erwin

Award date:
2022

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Industrial Engineering & Innovation Sciences
Information Systems Research Group

Master thesis *Erwin Martens* *student ID 1495798*

Robust & decentralized project scheduling

Master's program Operations Management & Logistics
Special track Manufacturing Systems Engineering

First	*TU/e supervisor*	*dr.ir. H. Eshuis*
Second	*TU/e supervisor*	*dr. Q.V. Dang*
Third	*TU/e supervisor*	*dr. H.J.S. Baier*
First	*company supervisor*	*ing. R.A. Dijkstra*
Second	*company supervisor*	*col. F. van Wieren*

March 6, 2022

Abstract

The IT-division of the Dutch Ministry of Defence (MinDef) postpones the majority of due dates of its projects in Change Management (CM) and Project Management (PM), reducing in-time delivery and throughput. The cause is a resource allocation and project scheduling strategy that does not account for uncertain and asymmetrically distributed information. To solve this Resource Constrained Multi-Project Scheduling Problem (RCMPSP), a Robust & Decentralized Project Scheduling Method (RDPSM) is designed using an existing Multi-Agent System (MAS) as core framework which is modified and extended to improve quality robustness. The method is validated using a Monte Carlo simulation and the MPSPLib benchmark dataset. Results indicate that the method improves quality robustness of schedules compared to the original method without robustness modification.

Keywords: *Resource Constrained Multi-Project Scheduling Problem*, *Decentralized*, *Robust*, *Multi-Agent System*, *Combinatorial Auction*, *Combined Slack Sufficiency*

Summary

In this master thesis, a Robust & Decentralized Project Scheduling Method (RDPSM) is developed for the IT-division at the Dutch Ministry of Defence (MinDef). The IT-division, Joint Informatievoorziening Commando (JIVC), fulfills IT-project orders from other branches at MinDef. Management at JIVC has a problem managing and controlling projects in CM and PM. A majority of project due dates are postponed, leading to increased makespans and decreased in-time delivery (reliability). Management wants to improve on these two aspects.

Projects are scheduled, making it a Resource Constrained Multi-Project Scheduling Problem (RCMPSP). JIVC is unable to execute work according to their manually made schedules, leading to postponed due dates. The main cause lies in three assumptions in current resource allocation and project scheduling. The assumptions do not account for uncertainty, disruptions and asymmetrically distributed information. The latter is the separation of information across DMUs in JIVC. Hence, the current strategy does not account for uncertainties, disruptions or asymmetric information distribution, making the current resource allocation and project scheduling strategy is non-optimal.

Decentralized project scheduling accounts for asymmetrically distributed information and focuses on improving resource allocation. Robust project scheduling accounts for uncertainty and disruptions in project scheduling problems and focuses on improving project scheduling. Hence, the current non-optimal strategy can be improved by employing a project scheduling method that is both robust & decentralized. Unfortunately, to the best knowledge of the author, there does not exist a robust & decentralized project scheduling method (Martens, 2021). Therefore, one is designed in this thesis to enable JIVC to improve reliability and makespan through project scheduling and resource allocation.

The design of such as method is not self-evident. The robust methods that solve RCMPSPs make use of information that is regarded private in a decentralized method. Hence, those methods cannot be employed. An alternative is to improve robustness of projects individually, yet literature indicates that disregarding the multi-project setting in robust project scheduling is non-optimal. The thesis builds on solving this problem through the design of RDPSM.

Preliminary knowledge is presented regarding both robust and decentralized project scheduling to guide in the search for suitable methods. The classification of W. Herroelen and Leus (2005) is used, being the five types: pro-active, reactive, stochastic, and fuzzy project scheduling, and sensitivity analysis. Decentralized scheduling methods generally employ a Multi-Agent System (MAS) as a core framework. In it, agents represent DMUs that interact, and these interactions are limited by most (combinatorial) auctions or negotiations interaction protocols.

The CM and PM processes at JIVC are analysed to determine 1) the data in the as-is situation and 2) how an RDPSM could be implemented in that processes and 3) the most common kind of uncertainty at JIVC. The conclusion is that there is a lack of suitable data in the current processes to construct a problem instance from existing information systems and databases. Management decided to continue the development of the proposed RDPSM. An alternative dataset is used as problem instances, being the MPSPLib dataset (http://www.mpsplib.com/). Management of JIVC indicates to focus on the following uncertainty: 10% of tasks in a project are delayed by 10%.

The organizational hierarchy of JIVC is analysed to determine how agents can be modeled in the MAS. Projects are allocated to entities within JIVC that become responsible for the execution of that order. These entities are represented by project agents that all own one project that want to minimize makespan. Conflicts arise between these agents for resources. The employees in resource pools within the sections are identified as resource agents, that aim to level resource usage. Finally, the management of JIVC is represented by a mediator agent that aims to choose what is best for the organisation as a whole (maximizing social welfare or strategic value).

Adding robustness should increase in-time delivery (reliability) at JIVC, and measures are required to assess the robustness of a schedule and performance of the method. Robustness is in general the ability to absorb uncertainty. One of the most recognized views is that as provided by W. Herroelen and Leus (2005), which differentiates between stability and quality robustness. Stability robustness (or system nervousness) focuses on differences (distance) between the start times of all tasks before and after being subjected to uncertainty. Quality robustness focuses on the distance between due date and actual realization date after being subjected to uncertainty. Quality robustness is therefore more relevant compared to stability due to the aim to improve in-time delivery. Khemakhem and Chtourou (2013) experimented with surrogate robustness measures to determine which have the best correlation of indicating robustness and actual robustness. The highest correlating measure is the Combined Slack Sufficiency (CSS).

Suitable robust project scheduling methods are explored, and only pro-active project scheduling methods are relevant. Within this category, there are still many methods, and the publication by W. Herroelen and Leus (2007) is used to refine the search based on a level of variability and dependency. The advise for situations that match that of JIVC is to create "a robust drum plan" without too much detail in order to respond to uncertainties. E. Demeulemeester and Herroelen (2010) presents a collection of methods and four are potentially suitable. Two are time buffer insertion methods, being the Critical Chain Method (CCM) and STC, and the other two are resource flow network, being the Partial Order Schedule (POS) with chaining and Myopic Activity-Based Optimization (MABO). No method is chosen since the decision depends on the MAS of the decentralized method.

A decentralized method is required in the design of the RDPSM. Many synonyms are used in literature for decentralized systems, and not all methods are truly decentralized. Therefore, identifying suitable methods is difficult. The most common (and often truly decentralized methods) employ negotiations or Combinatorial Auction (CA)s. Therefore, an SLR is conducted to very precisely search for project scheduling methods that employ either of these two interaction protocols. Eight publications are identified. The publication of Song et al. (2016) is determined to be the most suitable. In it, a CA is organised

between project agents and a mediator agent.

An RDPSM is designed. In the core framework of Song et al. (2016), project agents place bids for resources, the mediator agent selects initial winners by determining the project that are most beneficial according to a measure, after which project agents are allowed to modify their bids. Finally, the mediator agents selects final winners based on the modified bids. This framework is replicated in Python, including a custom parser, resource agents, and output logs and plots for managerial purposes. This replication is the first step towards an RDPSM, yet the replication does not have a robustness improvement step. The four robust methods previously identified are considered, but none are suitable for implementation. The reason is a new problem dynamic, being the trade-off when considering the amount of resources to select when adding slack. Robustness is therefore improved by designing a custom buffer insertion algorithm inspired by the STC approach. It employs the surrogate CSS measure, and slack is added to tasks that have the highest approximate gain of that measure. The replication is modified such that robustness improvement is improved in each bidding and bid modification round.

The RDPSM is validated in two ways. The first compares the generated schedules between those reported in the publication of Song et al. (2016), the replication without robustness modification and the RDPSM. The first conclusion is that the publication of song is not perfectly replicated. The average project duration of the replication is on average 13,1% longer compared to the original publication, which is unexpected as these should be (approximately) equal. The second conclusion is that the RDPSM increases average makespan (5,0%) and total makespan (4,9%) and average project delay (13,9%) compared to the replication, which is not unexpected since robustness is traded for time.

The second comparison examines schedules after being disrupted to uncertainty. A scenario generator is created that generates a scenario of disruptions. The schedules generated by the replicated version and the RDPSM are both subjected to scenarios to compare differences in in-time delivery and makespan. The in-time delivery performance of the replication is on average 30,5%, compared to 52,7% of the RDPSM. Hence, the first conclusion is that RDPSM schedules are on average 22,2% more robust (a 73,0% increase). The second conclusion is that this improvement comes at a cost of on average 15 days (a % 5,4 increase). Looking in more detail to the experiment results it can be concluded that some instance characteristics have a significant impact on the effectiveness of the method, such as resource constrainedness. For some instances, the in-time delivery only increases by 0,80% whereas others increase by 46,52%. The performance may be improved by fine-tuning algorithm parameter configuration, possibly based on the problem characteristics. Furthermore, three implementation plans are provided for JIVC, one for manual as-is use of the RDPSM, one for minimal implementation with some automated processing and one for a full implementation.

The overall conclusion is that the RDPSM improves robustness but does not guarantee in-time delivery. It achieves its main goal by improving robustness of schedules without violating information privacy and shows that this is a feasible approach. Due dates are still postponed in on average 47,3% of the cases, but this heavily depends on problem characteristics and ranges in the test set from 94,5% in the worst case to 2,3% in the best case. These characteristics, as the resource constrainedness, can be efficiently measured up-front and serve as an estimate for the instance performance. Hence, the designed RDPSM could support JIVC in improving in-time delivery and makespan through a more

optimal resource allocation and project scheduling strategy. Contributions to the field of project scheduling is the integration of decentralized and robust project scheduling methods and the accompanying problem dynamic, being the trade-off in determining the amount of resources to reserve when adding slack. Further research could be devoted to use the RDPSM in an actual case study, evaluation of the devised algorithm under certain instance characteristics, and investigation of the effects of the different amounts of resources to add in slack.

Acknowledgements

During the past two years, the corona pandemic effected the global population. The significance of its effects on my education cannot be overstated, therefore it feels right to mention personal acknowledgements first. I am grateful for my parents and my brother for their support (whether spoken, unspoken or in a material sense) during my pre-master and master education. Furthermore, I thank my friends for their frequent visits and talks which indirectly helped me. I thank my fellow pre-master peers for their perseverance throughout our studies, without them I would not have managed to graduate.

I thank my former mentor dr. M. Firat for all our constructive talks and his educational guidance during the start of my master till his transfer. Subsequently, I thank my subsequent mentor dr.ir. H. Eshuis for his guidance through the latter part of the thesis and his flexibility in planning the reviews of my thesis. Furthermore, I thank dr. Q.V. Dang for his motivating enthusiasm for the thesis topic in our conversations. I thank my company supervisor Richard Dijkstra for the unprecedented freedom I received, I could hardly have wished for a better supervisor. I thank all colleagues at MinDef who were part of my research, which are too many to name. Finally, I thank those who I may by mistake have forgotten to mention.

- *Erwin Martens*

March 6, 2022

Contents

List of Figures

List of Tables

Acronyms

AMS Average Makespan. 95

APD Average Project Delay. 61, 62, 95

CPD Critical Path Duration. 91, 94

CPD_{max} Portfolio Critical Path Duration. 94

CSS Combined Slack Sufficiency. iii, iv, 36, 38, 45, 54–57, 60, 63–65, 77, 78

DPD Standard Deviation of the Project Delay. 95

MS Makespan. 93

$NARLF'$ Normalised Average Resource Loading Factor'. 92, 94

NC Network Complexity. 92

PD Project Delay. 93

$PDEL$ portfolio delay. 95

RG Relative Gap. 93

RU_k Resource Utilization. 93

SD Start Delay. 95

TMS Total Makespan. 95

UF Overload Factor. 91

UF_k Utilisation Factor. 91, 93, 94

σ^2_{UF} Variance of the Utilisation Factor. 92

ACO Ant Colony Optimization. 123

AON Activity-On-Node. 91

CA Combinatorial Auction. iii, 21, 41, 42, 45, 76, 103, 115, 118, 123–125

CBO Colliding Body Optimization. 118

CC Critical Chain. 40

CCM Critical Chain Method. iii, 40, 53, 74, 112, 119

CI Confidence Interval. 148, 156

CLT Central Limit Theory. 156

CM Change Management. i–iii, 3–5, 12, 22–24, 26, 27, 30, 72, 120

CNP Contract Net Protocol. 21

CP Critical Path. 40, 91, 112

CSS Charged System Search. 118

DAG Directed Acyclic Graph. 91

DE Differential Evolution. 118

DMO Defence Materiel Organisation. 1

DMU Decision Making Unit. ii, xvii, 3, 5, 9, 16, 19, 21, 32, 50, 59, 74, 78, 98–100

DRCMPSP Decentralized Resource Constrained Multi-Project Scheduling Problem. 8, 9, 42, 60, 78, 123

e-CF e-Competence Framework. 22, 23, 32

FIS Fuzzy Inference System. 112, 123

GA Genetic Algorithm. 8, 118, 120, 123

GII Generieke IT & Infrastructuur. 1–3, 12, 22–24, 26

ITIL Information Technology Infrastructure Library. 24, 25

JIVC Joint Informatievoorziening Commando. ii–iv, x, 1–14, 16, 17, 22–27, 29, 31–33, 35, 38–41, 44, 45, 53, 54, 59, 70–77, 98, 99, 112, 121

KPI Key Performance Indicator. 26, 27

MABO Myopic Activity-Based Optimization. iii, 41, 53

MAS Multi-Agent System. i–iii, 8–11, 19, 20, 29, 32, 34, 40, 46, 60, 76, 78, 100

MCDMP Multi-Criteria Decision Making Problem. 117, 119

MCNM Modified Contract Net Mechanism. 123, 124

MILP Mixed-Integer Linear Programming. 120

MinDef Ministry of Defence. i, ii, vi, 1, 2, 8, 12, 24, 27, 29, 31–33, 76, 78, 99

MPM multi-project-management. 3

NN Neural Network. 123

NPV Net Present Value. 36

PM Project Management. i–iii, 3–5, 12, 22–24, 26, 27, 30, 72

POS Partial Order Schedule. iii, 41

PPM project portfolio management. 3

PSO Particle Swarm Optimization. 53, 118, 120

PSP Project Scheduling Problem. 6, 7, 9, 11, 14, 19–22, 26, 27, 29, 33, 38, 39, 45, 50, 75, 90–93, 95–97, 100, 112, 117–121, 123

RCCP Rough Cut Capacity Planning. 93

RCMPSP Resource Constrained Multi-Project Scheduling Problem. i, ii, 1, 7, 8, 14, 19, 30, 51, 53, 59, 69, 70, 76, 94, 95, 104, 118, 123, 124

RCPSP Resource Constrained Project Scheduling Problem. 6, 14, 29, 36, 49, 51, 53, 56, 69, 75, 78, 90, 92–96, 104, 117–121

RDPSM Robust & Decentralized Project Scheduling Method. i–v, 1, 8–11, 33, 35, 37, 45, 46, 59–62, 65–79

RM Run Management. 3, 4, 23, 31

RSEM Root Square Error Method. 40

SA Simulated Annealing. 118, 120, 123

SGS Schedule Generation Scheme. 48, 67, 69, 78, 92, 95, 117, 118, 123, 124

SLR Systematic Literature Review. iii, 8, 11, 12, 35, 41, 45, 78, 112, 113

SPSP Software Project Scheduling Problem. 75, 78

STC Starting Time Criticality. iii, iv, 40, 41, 51, 54, 57, 60, 76, 77

TOC Theory Of Constraints. 40

WDP Winner Determination Problem. 20, 21, 100, 103

Glossary

asymmetric information distribution (or decentral information, local information, private information, partial information) Information is separated over multiple self interested decision making units that may or may not share information (truthfully). (Fink & Homberger, 2015). 5, 29

baseline schedule (or static schedule, offline schedule, pre-schedule, predictive schedule, resilient schedule, sometimes robust schedule) A schedule generated by static scheduling (Rasconi et al., 2008). 14, 17, 18

bid Expression for desired goods in an auction using monetary values.. 19

bidding language See interaction protocol.. 19

decentralized project scheduling A category of multi-project scheduling methods that distributes the problem into less complex sub-problems that are individually solvable while regarding information of the sub-problems as private. The following terms are regularly found in literature but it is unclear if these are truly decentralized or just distributed: distributed project scheduling, multi-agent project scheduling, agent-based project scheduling, population-based project scheduling, hierarchical scheduling. (Fink & Homberger, 2015). 8, 9, 19, 98–100

disruption (or event). A random event changing a parameter, variable or other aspect of a project scheduling instance that one becomes aware of at some epoch in time between the start of the time horizon and the occurrence of the event. 6, 15

distributed project scheduling A category of multi-project scheduling methods that distributes the problem into less complex sub-problems that are individually solvable without considering symmetric or asymmetric information distribution. Decentralized project scheduling is a subset within distributed project scheduling. (Fink & Homberger, 2015). 98, 99

dynamic scheduling (or online scheduling, schedule repairing, schedule fixing) A category of project scheduling methods that (partially) (re)generate schedules while the schedule is being executed. While time progresses, new information becomes available regarding uncertainties. These methods react to new information and new (partial) schedules are generated (Rasconi et al., 2008). 15, 18

flexibility Some measure (multiple mathematical definitions) that indicates to what degree a schedule can change the starting times of events, a degree of freedom of constraints in a schedule (Billaut et al., 2008). 16

fuzzy project scheduling Category of robust project scheduling methods that models uncertainty using Fuzzy Logic and Fuzzy Inference Systems (W. Herroelen & Leus, 2005). 18

interaction protocol (or bidding language) Definition of allowed interact between agents.. 19

offer Expression for desired goods in a negotiation.. 19

pro-active project scheduling (periodic scheduling, sometimes robust scheduling) Category of static and robust project scheduling methods that produce baseline schedules (W. Herroelen & Leus, 2005). 17

pro-active reactive project scheduling (or hybrid project scheduling, predictive-reactive project scheduling) Category of robust project scheduling methods that have both a static and dynamic component (W. Herroelen & Leus, 2005). 18

quality robustness Collection of robustness measures aiming at quality, which is assessing the performance (or degradation) of the objective function in an executed schedule that was subjected to uncertainty (W. Herroelen & Leus, 2005). 36, 37, 40

reactive project scheduling (or event-driven scheduling, real-time scheduling, rescheduling) Category of dynamic and robust project scheduling methods that produce fixed schedules (W. Herroelen & Leus, 2005). 18

repaired schedule (or dynamic schedule, fixed schedule, online schedule, partial schedule) A (partial) schedule generated by dynamic scheduling (Rasconi et al., 2008). 15, 18

robust project scheduling A category of project scheduling methods that aim to produce schedules being able to absorb some amount of expected uncertainty. These methods produce 'robust schedules' (W. Herroelen & Leus, 2005). 7–9, 11, 14, 15, 17

scheduling flexibility (or freedom) Indication of flexibility or freedom that a DMU has in (re)scheduling, such as temporal freedom or sequential freedom (Billaut et al., 2008). 16

sensitivity analysis Category of robust project scheduling methods that does not produce a schedule but analyses how a change of a parameter effects some measure of a schedule (W. Herroelen & Leus, 2005). 19

slack (or free slack, float). The number of discrete time epochs in which the starting time of a task can be changed without effecting the starting time of preceding or succeeding tasks. 15

stability (or system nervousness, start time sensitivity) Some measure (multiple mathematical definitions) that indicates to what degree actual starting times in an executed schedule that was subjected to uncertainty deviates from the scheduled starting times (W. Herroelen & Leus, 2005). 37

stability robustness (or solution robustness, solution space robustness). Collection of robustness measures aiming at stability (see stability) (W. Herroelen & Leus, 2005). 37, 41

static scheduling (or offline scheduling) A category of project scheduling methods that generate a schedule before the execution of that schedule has initiated. While time progresses, new information becomes available regarding uncertainties. These methods do not react to new information (Rasconi et al., 2008). 14, 17

stochastic project scheduling Category of robust project scheduling methods that uses known information regarding uncertainty in scheduling and uses probability distributions to model that uncertainty (W. Herroelen & Leus, 2005). 18

symmetric information distribution (or central information, global information, public information, full information) All information is available to a central decision making unit. (Fink & Homberger, 2015). 5, 98, 99

treatment Name of the solution proposed in this treatment (Wieringa, 2014). 11

Chapter 1

Introduction

In this master thesis, a Robust & Decentralized Project Scheduling Method (RDPSM) is developed for the IT-division at the Dutch Ministry of Defence (MinDef) that deals with a Resource Constrained Multi-Project Scheduling Problem (RCMPSP) subject to asymmetrically distributed and uncertain information. Five topics are introduced in this first chapter. First of all, the organization of MinDef is presented at which the thesis is conducted (section 1.1). Subsequently, the problem context in the department Joint Informatievoorziening Commando (JIVC) is discussed (section 1.2). Next, research questions (section 1.3) and research methodology (section 1.4) are stated. This chapter is concluded with the outline of the thesis (section 1.5).

1.1 Organization

The organisation is introduced in three parts. First of all, the structure of the organisation is presented to indicate at which department the thesis is conducted (subsection 1.1.1). Next, the general activities of the department JIVC are introduced (subsection 1.1.2). Finally, the hierarchical decision making structure is introduced, which indirectly dictates how activities are managed (subsection 1.1.3).

1.1.1 Structure

The organizational structure is presented in Figure 1.1.

MinDef is divided into branches (indicated in blue). Each branch is either operational or supportive. The three branches under the Chief of Defence and the branch Marechaussee are operational, whereas the other three are supportive. This thesis has been conducted within the branch Defence Materiel Organisation (DMO), which is the branch responsible for the procurement and maintenance of material, and IT.

Each branch consists of divisions (indicated in green), and DMO has nine divisions. Within DMO, this thesis has been conducted at the division JIVC, which serves all the IT-related needs of MinDef and employs about 3.500 people.

Each division consists of departments (indicated in red) and JIVC has seventeen departments, the majority being operational departments specialized in a certain IT-product category or products group for a specific defence branch. Generieke IT & Infrastructuur

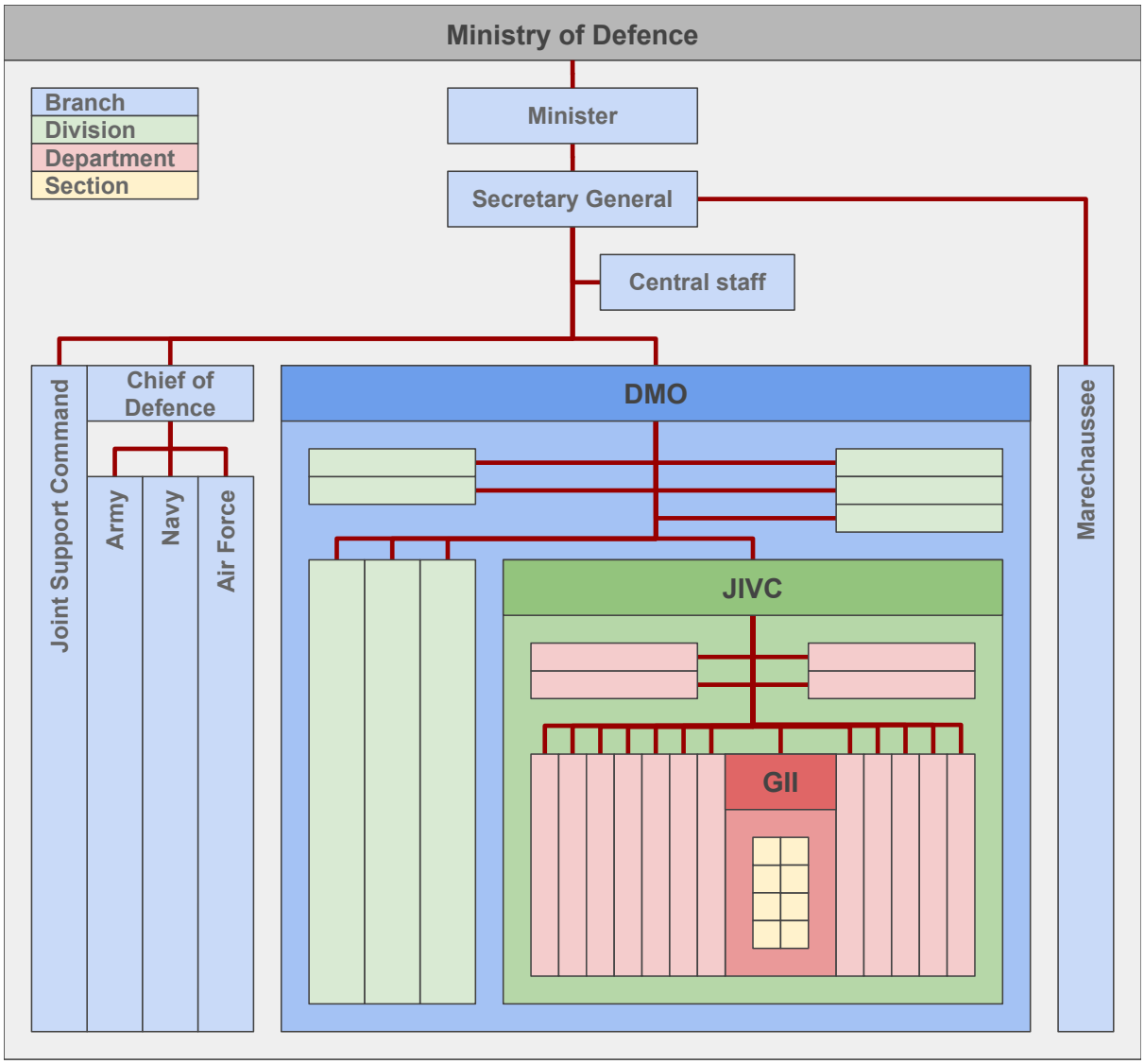


Figure 1.1: Partial organizational chart of the Ministry of Defence in 2021

(GII) is one of those departments, and responsible for generic IT products. It is by far the largest of the departments, employing about 1.200 people out of the 3.500 employed by JIVC. The thesis has been conducted at the department GII, the other departments are out of scope.

Each department consists of sections (indicated in yellow). GII is organized in eight sections. Six of them are specialized operational sections. Employees have roles as for example business analysts, system administrators, database engineers, system architects, solution designers, etc. The two remaining sections are supporting sections. The problem owner and secondary company supervisor is the former director of the department GII, and was the director at the initiation of the thesis. The primary company supervisor is the manager at one of the supporting sections within GII.

1.1.2 Activities

JIVC provides all IT services for MinDef. The 'clients' of JIVC are therefore branches within MinDef together with some external associated organizations (i.e. National Police

and Coast Guard). There is a customer-supplier relationship between those ordering clients and the IT-division JIVC. The clients have a certain demand which they express by placing 'orders' at JIVC and JIVC can supply clients by using its resources to realize these orders.

The resources that JIVC can use to supply demand are in the context of this thesis the employees and their available capacity. Other resources, such as hardware and equipment, are out of scope for this thesis. The resource capacity is managed and is the main limiting factor regarding the orders that can be fulfilled and completed. JIVC defines five capacity types in their capacity management (Figure 1.2).

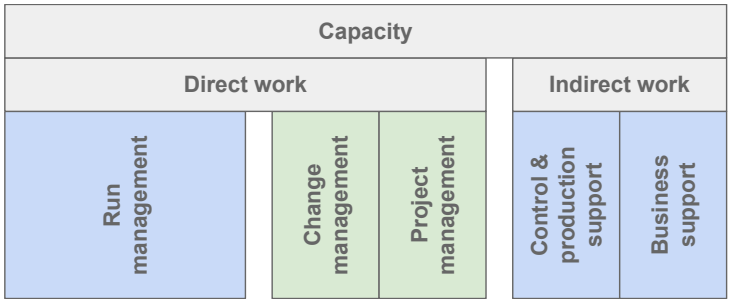


Figure 1.2: Five capacity categories as defined in JIVC capacity management

All five categories belong to either direct work or indirect work. In Q4 of 2021, about 84% of the capacity was spend on direct work whereas about 16% was spend on indirect work. Direct work contains three categories. The vast majority of the capacity is devoted to Run Management (RM), which keeps current operations running and includes activities as basic IT-maintenance and life cycle management. The second category is Change Management (CM), which processes changes to the current IT-environment that have limited and modest impact. The third category is Project Management (PM), which is the development of new products and services with substantial impact. Finally, there are two remaining categories belonging to indirect work, which are control & production support and business support. This thesis focuses on scheduling work in CM and PM (indicated in green).

1.1.3 Hierarchical planning and scheduling

A workload can be managed by employing resource allocation, planning and scheduling methods. These methods can be used by a Decision Making Unit (DMU) at different decision making levels. Most planning and scheduling models consider three decision making levels, as for example presented by Hans et al. (2007), E. Demeulemeester et al. (2007), and Joglekar et al. (2007). These three levels are strategical, tactical, and operational (Figure 1.3). JIVC employs such hierarchical planning and scheduling.

At a strategical level, executive and senior management make long term decisions in project portfolio management (PPM). JIVC prioritizes projects and composes project portfolio's. At a tactical level, project and resource managers decide on capacity and resource allocation in multi-project-management (MPM). JIVC creates a quarterly capacity plan in which resources are allocated for a certain workload. At an operational level, planners make detailed plans and schedules. At the sections of GII, there are decentralized planners making weekly plannings and taking care of day-to-day operations. As such, the

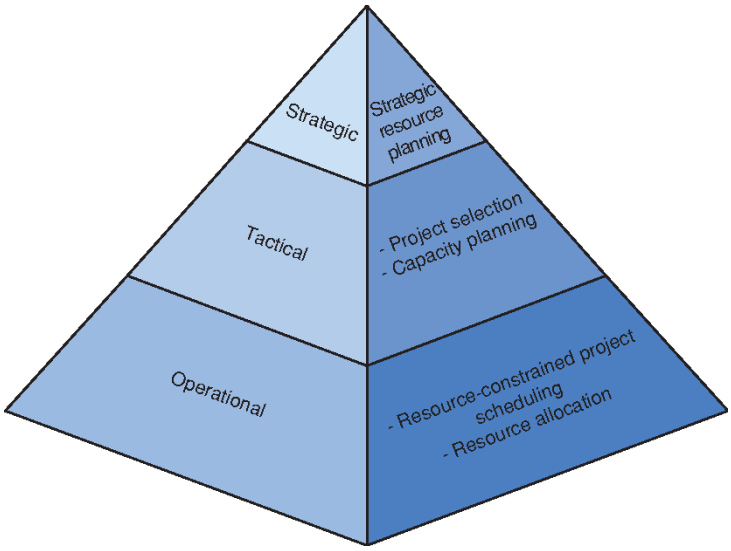


Figure 1.3: Levels of decision making in project management, from E. Demeulemeester et al. (2007)

three decision making levels are clearly distinct but still closely related, because decision making at one level influences decision making at the other levels (Pennypacker & Dye, 2002). This thesis focuses on the operational and tactical decision making levels.

1.2 Problem context

This section introduces the context of the problem of JIVC. First, the motivation for the research of this thesis is presented (subsection 1.2.1). Next, a cause and effect diagram is presented to pinpoint the exact causes of the problem (subsection 1.2.2). Finally, the business problem is stated, which is the main problem statement for this thesis (subsection 1.2.3).

1.2.1 Motivation

The direct motivation is the relative high percentage of due dates of the workload in CM and PM that are not met and therefore postponed. Management wants to increase the percentage of the workload that is completed before their respective due dates without postponing them. Therefore, this is a problem of managing and controlling the workload of CM and PM.

It is not surprising that the difficulty arises in particularly the capacity types CM and PM. The workload in the other capacity types (RM and indirect work) is characterized by being well predictable, having low variability and low uncertainty. Therefore, meeting due dates is not considered a problem for those capacity types. The workload in CM and PM though has a high degree of uncertainty and variability. Examples of the workload ranges from extending the capabilities of existing software, to research and development for the creation of novel software, to installing IT-equipment at a physical location and making that location operational. The lions share of workload is software development and configuration and new product development in an engineering to order regime. Therefore, it is not surprising that this workload is more difficult to manage and control. The scope

of the thesis is limited to these two capacity types.

1.2.2 Cause and effect

The observation that due dates are often surpassed in the workload of CM and PM is analysed in detail using a cause and effect diagram (Figure 1.4), starting with the problem (a).

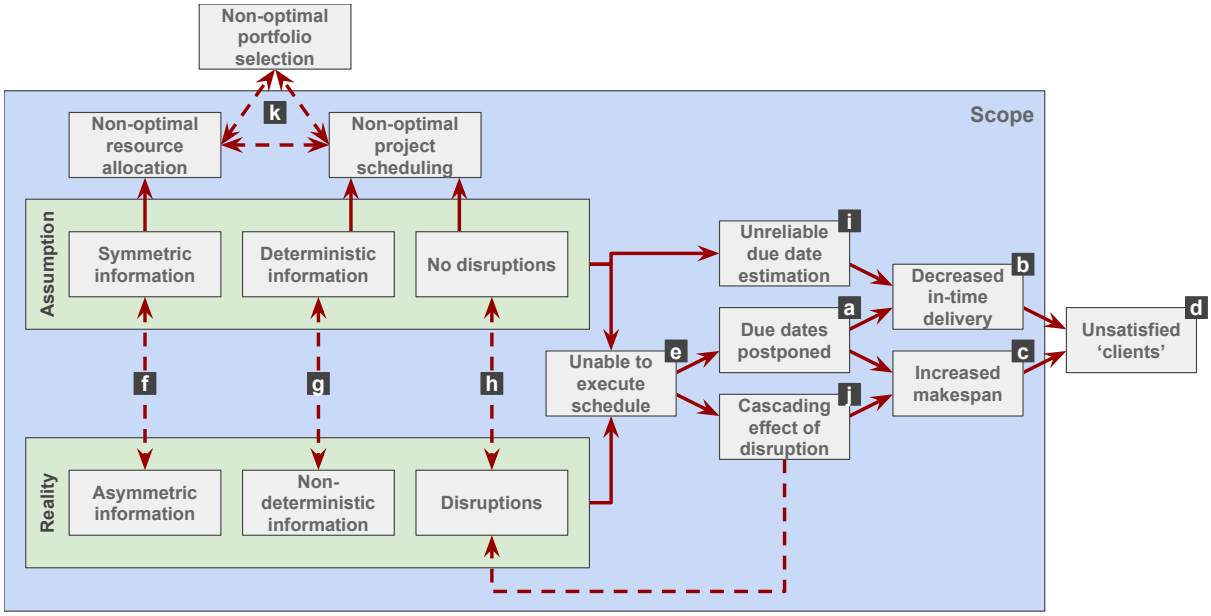


Figure 1.4: Cause and effect diagram of the problem at JIVC

Effects

First, the main effects of postponing due dates are presented. Starting at (a), due dates are often postponed. The first effect is that projects are not delivered within the intended due date (b) which degrades their in-time delivery reliability. Secondly, the project is postponed, increasing idle waiting time of tasks, increasing project makespan and decreasing overall project throughput (c). These two aspects (decreased reliability and throughput) result in a low customer satisfaction, which is what JIVC ultimately wants to improve (d). At this moment, there is no measurement for customer satisfaction, which is why it is considered out of scope for this thesis.

Causes

Due dates are postponed (a) because JIVC is unable to execute the schedule as planned (e). There reason lies in the assumptions being made in planning, scheduling and resource allocation that deviate from the actual situation. There are three main differences between the assumed and the actual situation.

The first assumption regards symmetric information distribution, whereas in reality, there is asymmetric information distribution (f). symmetric information distribution assumes a central DMU having access to all information (Fink & Homberger, 2015). In reality, there is asymmetric information distribution, meaning that information is distributed over multiple DMUs without anyone having access to all information. There are for

example multiple project managers, planners and resource managers who have partial, limited or indirect access to information and may additionally have personal goals. This wrong assumption deteriorates resource allocation, as also indicated by Hans et al. (2007). More detailed information regarding information distribution and a motivation why it is applicable at JIVC is presented in Appendix B.

The second assumption is that information is deterministic, while it is actually uncertain and vague (g). All project tasks have a forecasted an estimated resource requirement which are now regarded as deterministic, where uncertainty is barely accounted for. The third assumption is that disruptions will not occur, while disruptions are highly likely to occur (h). Disruptions are highly likely to occur due to for example high variation in task duration, unexpected drops in resource availability, changes in project priority, etc. Both the second and third assumption deteriorate project scheduling. Because of these difference in assumed and actual situation, JIVC is unable to execute its planned schedule.

Due to these assumptions, JIVC is estimating internal and external due dates incorrectly (i), which decreases in-time delivery (b). JIVC does acknowledge difficulty in quoting due dates. Another effect is that a single disruption effects all the work scheduled for that resource and all the work scheduled for that project at other resources, leading to a snowball effect (j). This cascading effect is essentially a positive feedback loop that causes new disruptions and repeat the loop.

The first assumption (f) leads to less optimal resource allocation, whereas the other assumptions (g, h) lead to less optimal project scheduling. As indicated in subsection 1.1.3, scheduling at an operational level is closely related to resource allocation at a tactical level, which is in turn related to project portfolio selection at a strategic level (k). Literature emphasizes how operational and tactical decision making are related, Hans et al. (2007) for example underline the importance of quoting tight and reliable due dates. Due to the intertwined relations, all aspects must be non-optimal. To keep the scope of the thesis manageable, strategical decision making is out of scope. However, insight at a tactical level may still benefit strategical decision making.

1.2.3 Business problem

To conclude, the business problem is stated as follows:

Business problem: The problem is non-optimal resource allocation and project scheduling of the workload in change management and project management based on invalid assumptions leading to a reduced in-time delivery (unreliability) and longer makespans.

The workload is in this context is managed and controlled through resource allocation and project scheduling. This problem is an optimization problem belonging to the category of resource allocation and Project Scheduling Problem (PSP). Background information on the classical PSPs is presented in Appendix A. Given limited resources, the problem is to schedule the starting times of projects and their tasks such that an objective is optimized. That objective is often the minimization of the makespan. Given constraints on resources, the problem is the RCPSP (see also section A.1). The problem of scheduling

a portfolio of projects using limited resources is referred to as the Resource Constrained Multi-Project Scheduling Problem (RCMPSP) (see section A.2). The classic RCMPSP is simple and not suited for most practical situations, such as the one encountered by JIVC. Therefore, researchers developed numerous generalizations and extension to deal with specific problems that practitioners encountered (section A.3).

1.3 Objectives and research questions

The previous section states the problem context and the business problem. This section first states two objectives that aim to solve that business problem (subsection 1.3.1). Next, the approach is explained on how to attain these objectives by stating research questions (subsection 1.3.2).

1.3.1 Research objective and design objective

The current assumptions as stated in the cause & effect diagram lead to non-optimal resource allocation and scheduling. The reasoning is that changing the invalid assumptions such that they take into account the actual situation solves the problem by resulting in a more optimal resource allocation and project scheduling (Figure 1.6).

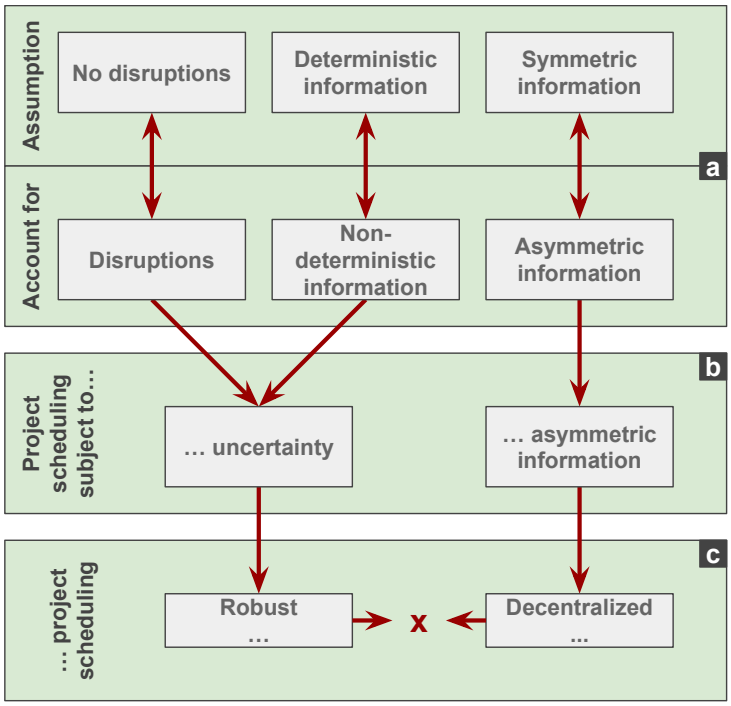


Figure 1.5: Approach

Three invalid assumptions are made in the current situation that should be accounted for in the desired situation (a). Researchers have studied all three aspects individually, two aspects belong to PSPs subject to uncertainty whereas the third aspect belongs to PSPs subject to asymmetrically distributed information (b). Researchers developed solution methods that solve these particular PSPs (c). Solution methods that solve PSPs subject to uncertainty are 'robust project scheduling' methods, most tailored to handle some kind of specific uncertainty in a single project setting. Solution methods that solve PSPs subject

to asymmetrically distributed information are 'decentralized project scheduling' methods. The specific problem is named the Decentralized Resource Constrained Multi-Project Scheduling Problem (DRCMPSP) in literature and always requires a multi-project setting. The most straightforward method to improve the current non-optimal scheduling strategy is to implement an RCMPSP that is both robust and decentralized.

Unfortunately, in a Systematic Literature Review (SLR) by Martens (2021) it was found that an RCMPSP that is simultaneously robust and decentralized such that it can deal with uncertain and asymmetrically distributed information of JIVC does not yet exist. Only in the past few years, researchers have attempted to develop a combined method that regards both problem aspects.

Simply combining an existing decentralized project scheduling method and a robust project scheduling method is not self-evident. Decentralized project scheduling methods enforce that information is private and cannot be shared between projects. Meanwhile, robust project scheduling methods in a multi-project environment require information of other projects to be able to absorb uncertainty effectively. This is a gap in the literature where this thesis will focus on. An RDPSM is to the best knowledge of the author novel.

Another limitation is set by MinDef, which is the use of a deterministic solution method. This means that a method always produces the same output and results given a unique problem instance and parameter configuration. This enables reproducibility, something that MinDef requires. This excludes a significant part of available solution methods since many heuristics employ a component using random generation (such as Genetic Algorithm (GA)s).

The scientific contribution lies in the research objective, stated as follows:

Research objective: Develop an RDPSM to solve a DRCMPSP subject to uncertainty and asymmetrically distributed information to increase in-time delivery while minimizing the makespan.

The corresponding design objective is stated as:

Design objective: Improve the in-time delivery and makespan of the current non-optimal scheduling strategy of JIVC by designing a MAS which solves a RCMPSP that takes uncertain and asymmetrically distributed information into account.

A RDPSM is valuable for practitioners dealing with both uncertain and asymmetrically distributed information. As indicated by Hazır and Ulusoy (2020), virtually all project scheduling instances are subject to some degree of uncertainty. Furthermore, most project scheduling situations deal with asymmetrically distributed information, meaning that self-interested decision makers are involved that may not share information truthfully. Therefore, the pool of practitioners in relevant situations may be substantial. Hence, designing a RDPSM for this problem advances knowledge in project scheduling and contributes to the scientific community.

1.3.2 Research questions

To accomplish the design objective, the problem is split into research questions. Before stating the research questions, two topics are introduced, being prior scientific research on the design of robust project scheduling methods and the role of a Multi-Agent System (MAS) in decentralized project scheduling methods.

Billaut et al. (2008) propose a three-step solution process of scheduling problems subject to uncertainties. The first step contains a definition and specification of the problem before execution or introduction of uncertainties, a specification and modelling of the uncertainties, and a specification of the quality aspects. The second step is the calculation of feasible solutions given the situation described in the first step. The third step is the execution of a schedule where it is subjected to uncertainty and where the schedule may be maintained or disrupted and rescheduled. This solution process is used to state research questions.

A central concept in decentralized project scheduling is a MAS. A MAS is a virtual environment consisting of self interested autonomous agents which represent self-acting DMUs. The idea is to decouple a large complex problem into smaller sub-problems over multiple agents, because solving sub-problems in a distributed system can be more efficiently. Fink and Homberger (2015) presents an introduction to and review of decentralized project scheduling, explaining how a MAS is the core framework to solve DRCMPSPs. The RDPSM contains a decentralized part, therefore implying that a MAS is required as its framework. A MAS is a versatile framework that is fully customizable, for example through agent representation and protocols for agent interaction. The setup of the MAS is described in the research questions.

Figure 1.6 presents the relation between research questions and the organisation of the thesis.

After presenting preliminary knowledge on both robust project scheduling and decentralized project scheduling in chapter 2, two research questions are answered in chapter 3 that investigate the problem and analyse the current situation at JIVC.

RQ1: How can data at JIVC be acquired to create project scheduling problem instances and what kind of uncertainty is encountered?

This questions answers the first step as described by Billaut et al. (2008). A PSP method requires a problem instance as input. Data at JIVC is not readily available for usage, moreover, data is scattered over multiple software suites and databases. Therefore, it is necessary to analyse processes and assess which data can be provided, determine the quality of that data, how the data can be extracted and what extraction effort is required. Furthermore, the type of uncertainty the schedule is subjected to must be determined.

RQ2: How to model the agents in the MAS such that it represents the structure and hierarchical decision making of processes at JIVC?

The MAS models agents, where agents are entities with objectives to represent actual decision making. Therefore, it is necessary to investigate what suitable structures are,

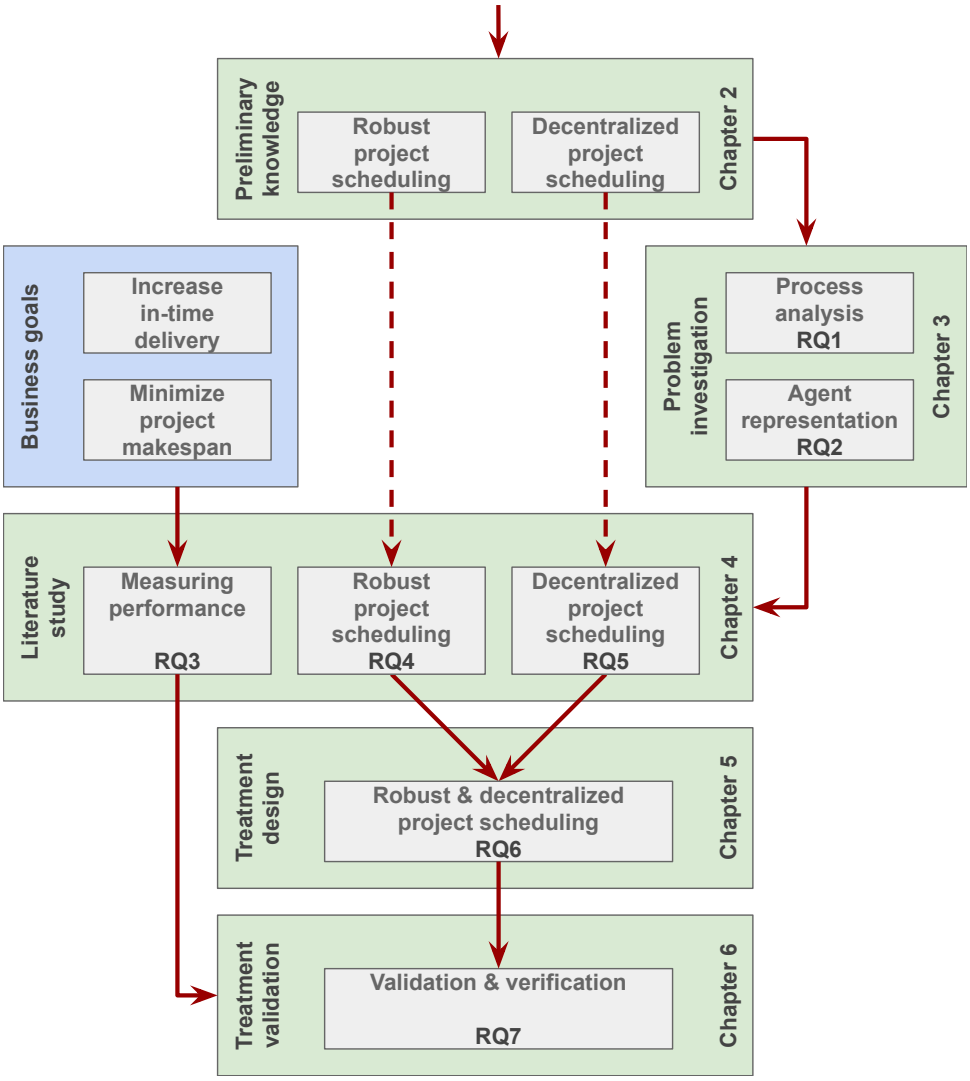


Figure 1.6: Organisation of research questions

what the goals and restriction are of the involved entities, and how these can be formulated as objectives and constraints in the MAS.

Subsequently, three research questions are answered in chapter 4 that all require the study of literature and the information collected in chapter 2 and chapter 3.

RQ3: How to measure the quality of a project scheduling problem solution and the performance of a RDPSM?

Measurements are required to quantify current performance and measure improvements of a new solution method. The business objective states two goals, and using the analysis of uncertainty at JIVC in chapter 2, proper measurements are explored and selected to measure performance of a RDPSM. Other measures are presented in Appendix A.

RQ4: What are the most suitable robust project scheduling methods that deal with the problem instance and uncertainty as encountered by JIVC?

There are numerous sources of uncertainty and a multitude of robust project scheduling methods. Using the analysis of uncertainty at JIVC in chapter 3, robust scheduling methods are selected that are expected to improve project scheduling. These are selected as candidates to use in a RDPSM.

RQ5: What is the most suitable multi-agent system that is truly decentralized, deals with the problem instance and asymmetrically distributed data as encountered by JIVC?

There are numerous ways to set up a MAS and tailor it to specific needs. One of the most important aspects is a interaction protocol. There are only two interaction protocols for MASs that solve PSPs and are truly decentralized, being the auction and negotiation protocols. Both are explained in the next chapter as preliminary knowledge. An SLR is conducted to find suitable MASs that are expected to improve resource allocation, after which the most suited one is selected.

In chapter 5, the methods explored in chapter 4 are used to answer the following research question.

RQ6: How to design a RDPSM that deals with the uncertain and asymmetrically distrubed information at JIVC?

A RDPSM is designed using a MAS with an auction or negotiation method chosen in ***RQ5*** and one or more robust scheduling methods from the selection identified in ***RQ4*** that is compatible with the selected MAS. This should improve the business objective of JIVC.

In chapter 6, the RDPSM designed in chapter 5 is used to answer the following research question.

RQ7: What is the performance of the RDPSM?

The project scheduling method should provide schedules that improve in-time delivery and minimize makespan, which is measured by the measures as identified in ***RQ3***.

1.4 Research methodology

In this section, a research methodology is chosen to answer the research questions of the previous section. This thesis is executed within the research group Information Systems and the RDPSM designed in this thesis can be regarded as a piece of software. Therefore, the research methodology for this thesis is based on the design science methodology for information systems and software engineering by Wieringa (2014). The solution to the problem is named the treatment. The core of the methodology is the design cycle, which is equal to the engineering cycle (Figure 1.7) except that the top-left phase 'treatment implementation' is skipped.

The first phase, the problem investigation, is split in two chapters. The first is chapter 3, focusing on investigating the as-is situation at the company to answer ***RQ1*** and ***RQ2***.

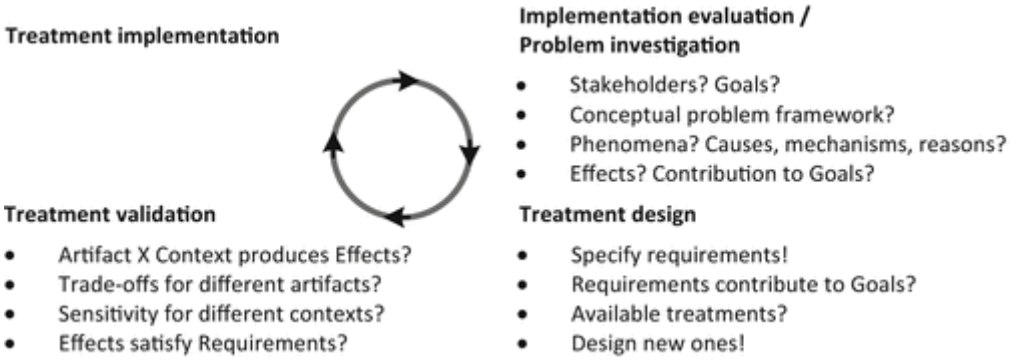


Figure 1.7: The engineering cycle. The question marks indicate knowledge questions, and the exclamation marks indicate design problems, from Wieringa (2014) (see page 28)

RQ1 analysis the process and is answered by desktop research and interviews. Over two dozen employees are interviewed that work at positions related to project scheduling, resource allocation, order processing, planning, CM, PM, etc. These are unstructured interviews due to the diversity of employees and their roles. Further information is acquired from the MinDef intranet. ***RQ2*** investigates agent representation and is answered by desktop research and interviews. The main desktop research is linked to organizational charts of JIVC, GII and its sections. The interviews are unstructured interviews with employees from a demand side perspective (customers, project managers), supply side perspective (resource pools, sections), and management perspective to understand goals and objectives.

The second chapter that is part of the problem investigation is chapter 4, focusing on investigating literature to answer ***RQ3***, ***RQ4*** and ***RQ4***. ***RQ3*** is answered by desktop research, reviewing literature on measurements of performance in combination with the information from ***RQ1***. A majority of publications seen in the preliminaries and appendices on project scheduling refer to a handful of existing surveys and publications that are influential in the field of project scheduling, therefore no Systematic Literature Review (SLR) is required to answer the research question. ***RQ4*** is answered by desktop research, reviewing literature on robustness in combination with the information from ***RQ1***. Again, a majority of publications seen in the preliminaries and in previous work (Martens, 2021) refer to the same handful of existing surveys and publications, hence no Systematic Literature Review (SLR) is required to answer the research question. ***RQ5*** is answered by desktop research to find the most suitable decentralized project scheduling method. An SLR is conducted for a focused search.

Next is the second phase, the treatment design, which answers ***RQ6*** in chapter 5. ***RQ6*** is answered by using the most suitable method identified in ***RQ5*** as core framework and improving robustness using one of the identified methods in ***RQ4***.

The third and last phase, the treatment validation, answers ***RQ7*** in chapter 6. ***RQ7*** is answered by the performance measures identified in ***RQ3*** and applying it to the method designed in ***RQ6***. The results are compared to the same method but without the robustness improvement modifications.

1.5 Outline

This section concludes the introduction. The rest of this thesis is organized as follows. First, preliminary knowledge is presented (chapter 2). Next, the as-is situation of project scheduling at JIVC is investigated (chapter 3). Subsequently, literature is reviewed to close gaps in knowledge (chapter 4). Afterwards, a treatment is designed (chapter 5). Following up is the validation of the designed treatment (chapter 6). Finally, the thesis is concluded by discussing the research and design of this thesis.

Chapter 2

Preliminaries

The previous chapter introduces two specific categories of project scheduling methods, being robust project scheduling and decentralized project scheduling. This chapter presents preliminary knowledge on robust project scheduling (section 2.1) and decentralized project scheduling (section 2.2). Both topics are required for understanding and answering the research questions in the subsequent chapters.

2.1 Robust project scheduling

Robust project scheduling methods aim to absorb a certain type of uncertainty to a certain degree. In practice, project scheduling is virtually always subject to uncertainty (or non-deterministic information). Task durations are often based on forecasting and estimations, resource availability may drop unexpectedly, due dates and priorities may change and even complete tasks and precedence relations may change.

Robust project scheduling method have different properties compared to other PSPs. These properties are presented due to their reappearing importance throughout this thesis (subsection 2.1.1). A definition for types of uncertainty is presented which is required to identify types of uncertainty at JIVC in ***RQ1*** (subsection 2.1.2). Subsequently, a types of robust project scheduling methods are presented which are required to select appropriate robust scheduling methods in ***RQ4*** (subsection 2.1.3).

2.1.1 Properties

This section introduces a number of relevant properties, aspects and insights of robust project scheduling.

Static versus dynamic scheduling

Methods that only generate schedules based on information known before the execution of the schedule are static scheduling methods, and they generate baseline schedules. The classic RCPSP and RCMPSP assume deterministic information. As such, no changes are expected during execution. Assuming non-deterministic information, uncertainty is introduced. Now, another schedule might be generated to account for uncertainty, which is still static scheduling.

The situation changes during execution. Due to uncertainty, events might occur that change problem instance parameters, variables or other aspects. These disruptions spoil schedules such that they may become infeasible. Therefore, scheduling methods are required that repair schedules. These are dynamic scheduling methods, and they generate repaired schedules.

Increasing robustness through slack

Slack is the time window in which a task can be freely shifted without effecting succeeding tasks. Therefore, it gives opportunity to absorb uncertainty, making it the main approach to embed robustness in a schedule. Making a schedule more robust therefore equals adding slack at some positions, since that slack can absorb uncertainty. Adding slack to a task increases the duration of that task, and if it is on the critical path, it delays the starting time of succeeding activities, hence increasing the makespan of the project.

Trade-off problem and two level approach

Robustness is a trade-off problem. In essence, robustness can be added indefinitely. Adding robustness is adding slack, which degrades other performance measures such as time and costs. Therefore, robustness cannot be seen as the only objective in an optimization problem. A trade-off problem arises between robustness and other measures such as makespan. Most robust project scheduling methods employ consist of two levels. In the first level, a schedule is generated optimizing an objective function not related to robustness such as minimizing the makespan. In the second level, that solution is enhanced or optimized for robustness in which it must consider the value of the first objective function. In some methods, robustness may only be (attempted to be) improved without degrading that value, whereas other methods are allowed to degrade that value to some extent. In conclusion, solution methods require some kind of notion of a limit in this trade-off problem, indicating how much of an objective function may be offered to gain robustness.

Evaluation by simulation

To evaluate robustness, a robust schedule must be generated, after which that schedule must be tested in an environment subject to uncertainty. Evaluating schedules in real life scenarios would be inefficient, since it takes the complete time horizon to just test one scenario. Therefore, researchers use simulations such that many scenarios can be tested efficiently.

Creating all kinds of possible scenarios to test a schedule is also inefficient. Therefore, a generator is required to generate scenarios, where a scenario is a collection of changes due to uncertainty. In most practical situations, it is infeasible to test all possible scenarios, hence, stochastic simulations (or Monte Carlo simulations) are employed to randomly generate scenarios. The scenario generator in the simulation explicitly requires a formulation of uncertainty that it may introduce with bounds and limits on parameters of the uncertainty. Therefore, the performance of a robust solution method is always bound based on the scenario generator setting.

Disruption awareness

Disruption awareness as stated by Rasconi et al. (2008) is a crucial concept sometimes overlooked in robust project scheduling. It considers the moment in time that a type of uncertainty is known to a DMU. Suppose a task of a certain duration that will take longer in the end. When will this be known to the decision maker? If awareness occurs at the start of the project, more options are available to fix that schedule compared to awareness at the scheduled completion of that task. A later awareness reduces the flexibility in rescheduling, hence limiting its options.

2.1.2 Types of uncertainty

This section selects a classification of types of uncertainty from literature to identify the uncertainty at JIVC in ***RQ1***. Kouvelis and Yu (1997) present sources of uncertainty. Subsequently, W. Herroelen and Leus (2007) identified types of variability. Rasconi et al. (2008) provide an overview of disruptions that may occur, being: 1) the delay of an activity, 2) the increase of an activity duration, 3) the decrease of resource availability, 4) the addition or removal of an activity, 5) changes to successor/predecessor relations. Hazır and Ulusoy (2020) present a classification on sources of uncertainty in project scheduling (Figure 2.1). This overview is used for ***RQ2*** in chapter 2 to identify types of uncertainty at JIVC.

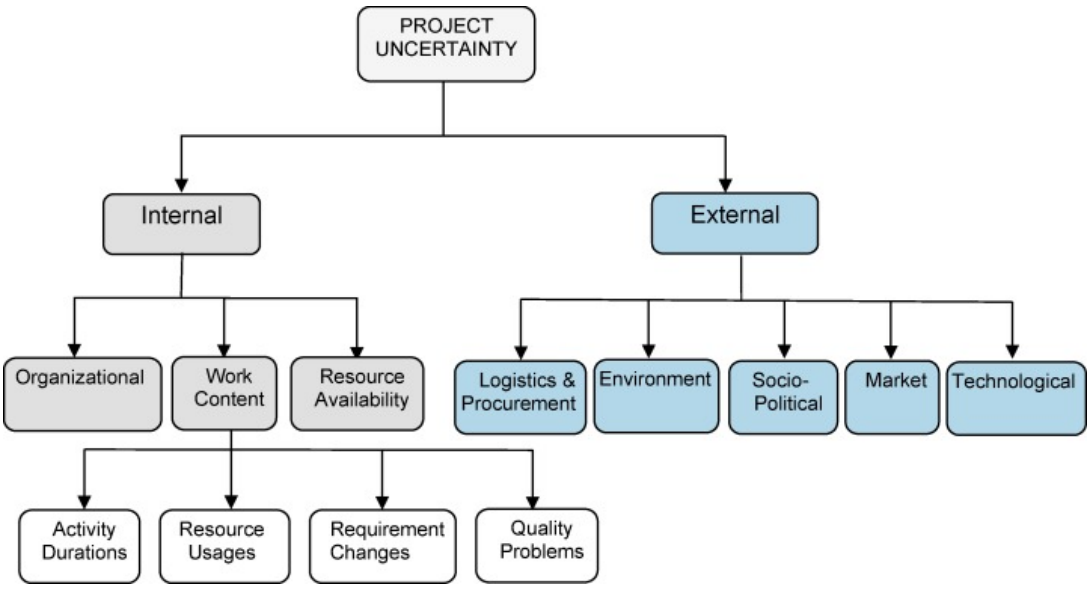


Figure 2.1: Project uncertainty sources and their classification, from Hazır and Ulusoy (2020)

Another perspective to consider is that of scheduling flexibility as introduced by Billaut et al. (2008). This concept indicates the freedom to change a schedule, for example as sequential flexibility or temporal flexibility. For example, there is no sequential freedom if a DMU is not allowed to change the sequence of tasks. This freedom limits scheduling flexibility and is therefore interesting to consider, since it may rule out solution methods.

2.1.3 Types of methods

This section selects a classification of robust project scheduling methods to identify to assist in selecting promising methods in ***RQ4***. Depending on the type or types of uncertainty at JIVC, some robust project scheduling methods may be irrelevant or ineffective. Therefore, the broad category of robust project schedulings methods is divided into subcategories. Multiple researchers provide classifications on methods for project scheduling subject to uncertainty, such as that of Davonport and Beck (2000). W. Herroelen and Leus (2005) provide a classification of uncertainty in project scheduling. That same group of researchers extended their work in a review of robust scheduling methods (W. Herroelen & Leus, 2007), an extended classification of robust scheduling methods (E. Demeulemeester & Herroelen, 2010) and the classification of predictive-reactive scheduling (Van de Vonder, Demeulemeester, et al., 2007). A more recent overview of robust project scheduling is provided by Chaari et al. (2014) and Ulusoy and Hazır (2021) (see section 12.2).

In this thesis, the classification as provided by W. Herroelen and Leus (2005) is used, due to its popularity in literature (Figure 2.2). It describes five types which are introduced next.

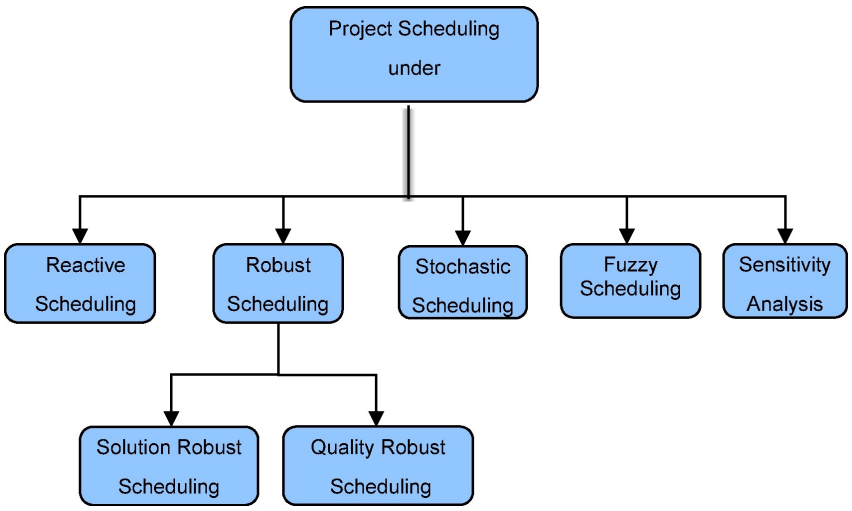


Figure 2.2: Taxonomy based on W. Herroelen and Leus (2005), from Hazır and Ulusoy (2020)

Type 1: Pro-active project scheduling

Pro-active project scheduling is a collection of static scheduling scheduling methods that generate a baseline schedule schedule to account for uncertainty before disruptions occur. Pro-active project scheduling attempts to attain a stable baseline schedule such that rescheduling costs are decreased and due dates are forecast reliably. According to Mehta and Uzsoy (1998), the baseline schedule serves two goals: 1) resource allocation and 2) external activity planning, such as activity and material preparation and delivery date forecasting. Wu et al. (1993) underlines the latter goal. The most common methods add time buffers to critical tasks that are prone to be delayed, generate a schedule were disruptions have the least probability of negative impact or generate a set of (fragmented) solutions to switch between during execution. As seen in Figure 2.2, there is a further division, which is related to measuring robustness and discussed in section 4.1.

Type 2: Reactive project scheduling

Reactive project scheduling is a collection of dynamic scheduling scheduling methods that attempt to fix or reschedule a schedule that is subjected to an unanticipated event during execution. The main goal for rescheduling is restoring precedence and resource feasibility. Fixing a schedule might not be possible if the impact of a disruption is to substantial. In that case, another type of method should generate a new baseline schedule.

pro-active reactive project scheduling

While not being stated as a type on it's own in the publication of W. Herroelen and Leus (2005), pro-active reactive project scheduling is hybrid. This is a separate bi-level setup, with a static primary level generating a baseline schedule prior to execution, and a dynamic scheduling secondary level during execution generating a repaired schedule. A hybrid is preferable above only a pro-active solution. There is a range of pro-active and reactive behaviour, as proposed by Rasconi et al. (2008) (Figure 2.3).

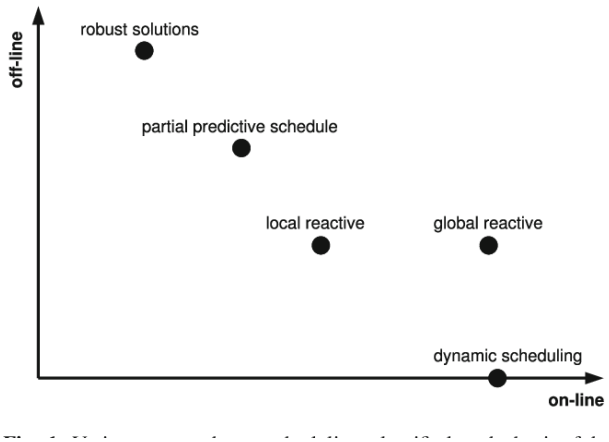


Figure 2.3: Various approaches to scheduling, classified on the basis of the off-line and on-line phases, from Rasconi et al. (2008)

Type 3: Stochastic project scheduling

In stochastic project scheduling, additional information regarding uncertainty and variability is taken into account when generating a schedule. Generally, probability distributions regarding task duration are employed in stochastic scheduling, therefore it requires information a priori. Stochastic project scheduling requires probability distributions to be known.

Type 4: Fuzzy project scheduling

Fuzzy project scheduling considers another type of uncertain information, being vagueness. It is different from other types of uncertainty from probability theory, such as stochastic variables. Vague processes may lack historical data, not have independent random variables, have unknown probability distributions or be highly uncertain.

Type 5: Sensitivity analysis

The last classification is sensitivity analysis, which studies the effect on the output based on changes to the input parameters. The research on this topic is sparse and mainly conducted for machine scheduling as indicated by W. Herroelen and Leus (2005).

2.2 Decentralized project scheduling

Decentralized project scheduling methods aim to solve a complex resource allocation and project scheduling problem in a distributed way while keeping some information private to certain agents. First, the general principles of a MAS that solves PSPs are presented (subsection 2.2.1). An understanding of interaction protocols is required to select suitable methods in ***RQ5***. These protocols are presented next, and are the negotiation (subsection 2.2.2) and auction (subsection 2.2.3).

2.2.1 Multi-Agent Systems that solve project scheduling problems

This section presents relevant principles of MAS that are used throughout the thesis. Additional background knowledge is presented in Appendix C

Market structure

The main goal of a decentralized RCMPSP is to allocate resources efficiently. Horling and Lesser (2004) describe types of multi-agent organisations. And of that is the 'market' type, in which an 'auction' is organized where there a supply side with sellers that offer goods to a demand side with buyers. Efficient resource allocation is one of the main advantages of a market structure. Therefore, the market is the classical model of a MAS that solves PSPs.

Protocols for interacting

The two common interaction structures are 'auctions' and 'negotiations'. In these structures, agents are limited in the information that they can share, by only being allowed to communicating through 'bids' or 'offers' respectively. Such a interaction protocol is known as the 'bidding language'. Bids and offers may be expressed in monetary values, which can be beneficial since it allows to compare values of different offers in a single comparable unit as prices. Additionally, agents can express the value of changing bids and offers by expressing them using 'payment costs' or 'side payments'.

Coordination and conflict resolving

Agents are self interested DMUs having individual objectives. These objectives are usually conflicting, such as bids or offers for the same goods. Therefore a global coordination mechanism is introduced to resolve issues between agents. The goal is to find Pareto efficient solutions where it is not possible to improve a local solution without worsening another local solution. These coordination mechanisms can be represented by introducing another agent. The agent considers what is best for everyone in stead of a single agent. What is best for everyone can be numerically calculated as 'social welfare' or 'fairness',

both terms do not have a single definition. The coordination and conflict resolving mechanism must furthermore decide which bids win, which is the Winner Determination Problem (WDP).

Winner Determination Problem

Determining a feasible combination of winning bid(s) is an allocation problem known as the WDP (Lehmann et al., 2013). The WDP is $\mathcal{NP}$-complete, hence, (meta-)heuristics are virtually always employed to find a solution within acceptable computational time and effort. How to solve a WDP is part of mechanism design and is MAS dependant.

Agent types

The classic setup of a MAS that solves PSPs has three agent types (Figure 2.4).

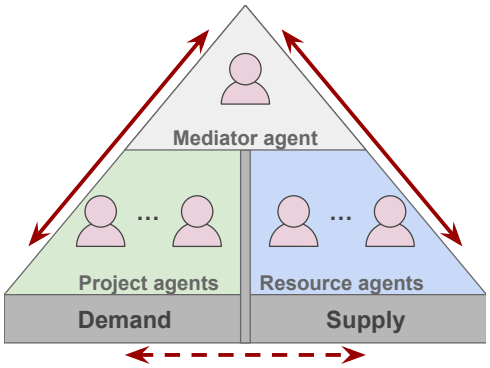


Figure 2.4: Classic agent types in a 'market type' multi-agent system that solves project scheduling problems

The demand side is represented by project agents, each representing a single project. The supply side is represented by resource agents, each representing one or more resources. The third agent a mediator agent or auctioneer that supervises the market and resolves conflicts. In distributed systems, all parties are allowed to access each others information, whereas in a decentralized system, information is private and cannot be accessed by other agents (see Appendix B). Interaction protocols limit the exchange of information between agents. There are multiple types of configurations for this interaction. In most protocols, project and resource agents do not interact directly with each other but only via the mediator, as indicated by the red arrows in the figure. In some configurations, there is limited interaction allowed between project and resource agents directly, or among project agents themselves. In yet other configurations, there are no resource agents, or just one resource agent, or no mediator agent.

Order of interaction

Wang et al. (2013) characterize three kinds of set-ups for a MAS in project scheduling, where the characterization depends on the order of interaction between agents. Most MASs have two layers, those for local scheduling at a project level and those for resource allocation at a coordination level. The first set-up is top-down, where resource allocation is first, succeeded with local project scheduling subject to the resource constraints. The second set-up is bottom-up, where projects are scheduled first and resources are allocated afterwards. The third set-up is peer-to-peer, which is in between the previous set-ups.

2.2.2 Negotiations

In negotiation, agents negotiate via an interaction protocol to come to an agreement. In this multi-round procedure, new and updated offers are generated and accepted or rejected. Due to the great freedom that negotiations inherently posses in expressing offers, there are numerous ways to implement negotiations. The communication between agents can be formalized in numerous ways. R. G. Smith (1980) developed the Contract Net Protocol (CNP), which is a versatile protocol and therefore popular as a starting point in multiple publications. In project scheduling, it is schedule related information that is being shared in the negotiation. Two common approaches are those that reveal (partial) schedules and those that reveal project sequences in project ordering lists. Fatima et al. (2014) produced a survey on negotiation models, whereas Vetschera (2013) published a book on the principles of automated negotiation.

2.2.3 Auctions

Auctions are special cases of negotiations, where offers are bids with a limited representation as prices in a monetary value. They are a structured way of negotiation abiding certain rules. At a market-oriented auction, goods are offered to bidders. Bidders can submit a bid for a good, based on value as perceived by the bidder. The goods are allocated to the highest bidder based on a pricing rule, after which the goods are allocated.

Goods in project scheduling

In a basic auction, an agent offers a single bid to acquire a single type of good. The goods offered in project scheduling are an amount of a resource that are sold per unit of time. Therefore, a bid contains a time-slot with an amount of a resource. Bidding at a maximum of one resource per round is not worthwhile in an auction for project scheduling. Suppose for example a project with two tasks, each requiring one unit of a different resource, where an agent wins a bid on one resource and loses the other. Winning only the first bid has no value since the project cannot be completed. Value is only obtained when all bids are won such that all types of resources are acquired to complete the total project. Therefore, it makes sense to bid on collections of goods.

This dynamic only occurs when DMUs can bid on a collection of goods. The value of a collection of goods may exceed or be less than the sum of the individual values. These are called complement goods and substitute goods respectively. The special type of auction where bids contain multiple goods is the Combinatorial Auction (CA). This introduces challenges in mechanism design through additional complexity in the bidding language, the WDP, expression of combined bids, information revelation, preference elicitation, and progression in CAs. One of the first and most influential publications of a solution method for solving a PSP using a CAs is that of Confessore et al. (2006). It is regularly used as a foundation in later works.

Chapter 3

Problem investigation

This chapter investigates the situation of project scheduling at JIVC to answer ***RQ1*** and ***RQ2***. The information in this chapter is collected through interviews with over two dozen employees in various positions throughout JIVC.

RQ1 aims to define a PSP instance at JIVC and to determine the uncertainty encountered at JIVC. A problem instance is defined by three pieces of information. The first one is the definition of all resource types (section 3.1), the second is the availability of these resources types at each epoch of the time horizon in consideration (section 3.2), and the third is the project data (workload) under consideration to be scheduled within that time horizon (section 3.3). The results of the analysis are presented next (section 3.4). Subsequently, the type of uncertainty encountered at JIVC is defined (section 3.5), which concludes ***RQ1***. Finally, ***RQ2*** is answered by representing the entities at JIVC as agents (section 3.6). This chapter is concluded with a summary of the answers to the research questions (section 3.7).

3.1 Resource types

The 1.200 employees at GII are hierarchically separated in departments, which are in turn separated in sections. Moreover, employees can be grouped based on three category labels. The first is the section that the employee works in. The second is is the main skill of the employee as defined in the European e-Competence Framework (e-CF) for ICT Professionals (EN 16234-1:2019). The third is a specialization of the main skill, all of which are custom subcategories defined by GII. These three labels form particular combinations that can be represented as tuples in the format (section, e-CF-profile, specialization). These combinations are called resource pools, and all employees belong to such a resource pool. In total, 216 unique combinations exist within GII at the time of writing. New specializations may be added to or removed from the list over time.

3.2 Resource availability

The capacity of JIVC to realize orders for CM and PM is limited by the availability of the resource pools, which is in turn determined by the availability of employees. Information of an employee's contract are stored in a software suite for HR purposes. It is not self

evident to determine the availability of resources or to estimate the expected availability. Multiple challenges arise. First of all, many different kinds of contracts exist, not only in the hours per week that an employee works, but also in that an employee can work for different departments or in different projects. Moreover, an employee might have multiple e-CF-profiles. On top of that, it is not specified how much time an employee can devote to work on orders. Recall that the lion's share of an employee's time is reserved to work on RM and that the remainder could be spend on CM and PM. This remaining percentage of time is the availability of an employee, which is different for each individual and variable over time, depending on many factors. Moreover, JIVC makes use of an additional external labour force to raise capacity or to make use of expert knowledge. These may be dedicated to certain projects and cannot be freely allocated. Therefore, due to all these uncertainties, the availability is an estimation that is mediocre at best.

Furthermore, JIVC has a capacity management office and GII has a capacity management board. For each section in GII, the availability of all employees is aggregated, after which an overall fixed percentage of the accumulated number of hours is reserved to work on orders. This accumulated time is in turn divided over the operational resource pools in that section according to relative availability. This is a gross estimate, because the hours of employees working in support are also taken into consideration in the aggregate and skew the percentages. The capacity of resources is used for decision making at a tactical level to determine quarterly capacity and resource allocation plans.

3.3 Project data

First, the main processes of orders is analysed to see how projects are created. Orders are submitted at a central order intake office (subsection 3.3.1). After the order intake, orders are processed either via CM (subsection 3.3.2) or PM (subsection 3.3.3).

3.3.1 Order intake

JIVC has a central order-intake office and uses a software suite that enables customers to submit orders in a predefined format. The simplified process of an order is presented (Figure 3.1).

The order-intake office reviews the characteristics of the orders on a weekly basis and categorizes those orders based on aspects such as the size, scope, workload, budget, risk and impact on the IT-environment (a). If orders contain insufficient information, they are returned to the customer who has to revise the order with additional information and resubmit it. If the order is sufficient, it will be delegated to the department within JIVC that has most affinity and commonality with that order, for example based on expertise, experience or domain knowledge. That department bears responsibility for the execution of the order.

First, small orders are filtered, which are directly executed using CM (b). They are aggregated with requests for change, which are small sized internal orders originating from other employees at JIVC. These requests for change bypass the order intake office since that office is focused on customers outside JIVC and not small internal requests. Small orders are characterized by being unpredictable and unforeseen, while they are often required to be executed in a short time window after order arrival. JIVC is not reserving

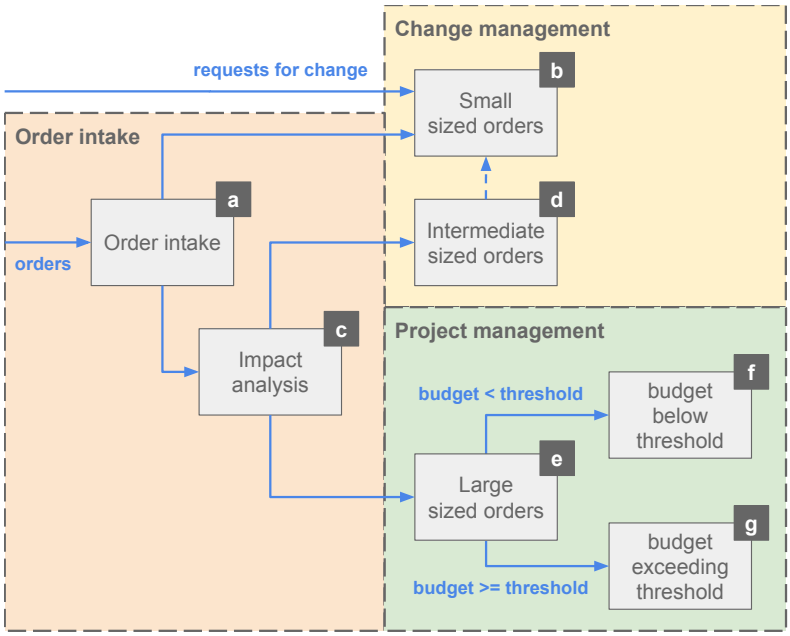


Figure 3.1: Flow of orders and relation between order types, change management and project management

capacity for small orders due to their unpredictable nature, event though it is virtually certain that small orders arrive. As such, small orders are executed ad-hoc by handling them on the fly in FIFO-order and scheduling them in the first suitable moment.

Back at the order intake office (a) all other orders are not small and require an additional impact analysis to determine if they are of an intermediate or large size (c). Intermediate orders are also executed using CM (d). JIVC can reserve capacity for intermediate orders, since these orders are more predictable and foreseen due to a prolonged order intake process. A typical intermediate order is divided into a number of smaller orders.

Large orders are executed using PM (e). There is a yearly budget for all projects in PM and a budget threshold per order. If an order exceeds the threshold, additional permission and authorization is required, resulting in orders that do not exceed (f) and do exceed (g) the aforementioned budget threshold. This mainly determines at which level decisions are made. Within the threshold, decision are made by a specialized IT-board overseeing all projects of JIVC. This board allocates projects to the department that is best suited to execute the project and therefore also allocates responsibility for execution. For projects exceeding the threshold, decision making lies even higher, exceeding that of JIVC. This is due to the strategic importance of those projects, the high budget involved and often the impact that the project has on MinDef.

As such, each incoming order is processed via either CM or PM, both processes are explained in the subsequent sections.

3.3.2 Change management and ITIL framework

Orders that result in a relative small or intermediate change to the IT-environment follow the Information Technology Infrastructure Library (ITIL) framework. The orders are allocated to a department (GII in the scope of this thesis), which becomes responsible

for the (timely) execution of the order. To manage order through the ITIL framework, a separate software suite is used. The resources required to complete these orders are financed by the budget of department. Recall that a department is subdivided into sections, and that some sections have a staff- or supporting-role whereas other sections have an operational role.

ITIL framework

The ITIL framework at JIVC is defined by six phases: 1) Initial, 2) Plan, 3) Assess, 4) Approve, 5) Realize, 6) Evaluate & close.

The initial phase (1) is started directly after allocation to a department by the order-intake office. The order arrives at a staff- or supporting section within the department which is now responsible for the execution for the order. A new order instance is created in the software suite used to manage the order, and that instance is linked to the original order in the other software suit using a unique key ID number. All information in this phase is already defined in the standard order format of the order-intake process, such as the goals of the project and a rough estimate of required resources.

Next, the supporting section creates a plan (2) that describes the work required to realize an order. This requires splitting the order into smaller tasks and an estimation of the workload for each task. The employees at this support section are not necessarily experts in the IT-domain, hence they may need to consult experts and professionals in the different operational sections in order to create a plan. Together with the inquired information of experts, the work to be done is estimated and a plan is created. The plan consists of a number of sequential tasks which are linked to specific resource pools. Furthermore, employees consult decentralized planners working in the operational sections to inquire when these resource pools are available in time to realize the tasks. The decentralized planners respond by allocating a timeslot for the completion. Now, each tasks is fitted in a timeslot and a due date is assigned to the task, which is saved via the software suite in the order instance. At this moment, it will be clear which section within the department has most affinity and commonality with that order. That section will later become responsible for realizing the order.

Before execution, the plan is assessed (3) to check how probable it is that execution will lead to achieving the goals as described in the first phase. Revision of the plan might be necessary. Next, the plan is up for approval (4). The order may be approved, delayed because it has low priority, or rejected altogether.

After approval, the plan can be realized (5) by the operational section it is allocated to. The decentralized planners within the section oversee the execution of the tasks within the order. The decentralized planner may allocate the task to an actual employee in the required resource pool. The sections is responsible for the execution of the tasks, even though that some tasks may be executed by other sections. For example, an order with a clear communications aspect is allocated to the communications section, but it also has a security aspect that has to be executed by the security section. Just as in any plan, changes may occur, which have to be acted upon ad hoc.

After completion, the order is closed and evaluated (6) by the supporting section in cooperation with the customer to check if the initial goals are achieved.

The instance object in the software suite registers in which phase the order is in. An employee completes a phase and the order progresses to the next phase. A timestamp is registered each time the data instance is mutated.

Data analysis

It is not registered in the software suite how long an employee actually worked on a task, phase or order. This would be valuable information for analysis. Knowing the service time, waiting time and throughput of orders, phases and tasks is valuable for analysis and may reveal important bottlenecks. For example, the throughput may be increased significantly if waiting times can be reduced, which therefore helps in reducing the company business problem. On the other hand, employees are required to register all hours worked and link those to the identification numbers of orders. Hours are registered in yet another software suite. Managerial insights are obtained after order completion by means of a subsequent calculation, comparing the estimated workload in to planning to number of hours actually worked on a project and the project tardiness/delay.

GII uses the number of instances completed in a month as KPI. At first glance, this may seem like a useful performance measure, but it is not that meaningful. Each instance is unique and may require many resources or almost none at all. A more meaningful performance measure would perhaps be to express workload of changes in hours and check how many work of the expected workload is being performed.

Tasks in CM are not directly linked to a resource pool. A resource pool (section, skill, specialization) is not a selection box but a description. Therefore, the entry in this field may not be accurate. Therefore it is difficult to create a PSP instance if tasks cannot be linked to resource pools they require.

3.3.3 Project management and stage-gate process

Orders that result in a relative large change to the IT-environment follow a stage-gate process and a combination of multiple PM tools. The orders are allocated to the project management department and a project manager is made responsible for the (timely) execution of an order. To manage orders in the process, a separate software suite is used. Decision makings related to these orders may be made outside of JIVC in case the budget threshold is exceeded, but execution of these projects is still the responsibility of JIVC and the departments in it. PRINCE2 project management method (https://www.axelos.com/certifications/propath/prince2-project-management) is used throughout PM.

Stage-gate process

Orders that result in a relative large change to the IT-environment follow a stage-gate process. At each stage-gate, an approval is required for continuation, otherwise the order may be halted, delayed or discontinued. The stage-gate process has the following stages: 1) Idea, 2) Analysis, 3) Initiation, 4a) Realisation, 4b) Implementation, and 5) Closure.

The deliverable of the idea phase (1) is actually the fixed format of an order to submit at the order-intake, therefore, the majority of the this phase precedes the actual order intake process. An order may not be valid at first submission, as such, the order-intake office

may contact the customer to give feedback and wait for resubmission of the order in an iterative process. Information of the expected activities is at this point in time rough and uncertain.

The second stage-gate phase is the analysis phase (2). The main deliverable is a project proposal. Very similar to CM, project managers consult departments, section and specialists to inquire expert opinions and information to create estimations of the workload required to realize the order. The information of expected activities is more detailed and certain compared to idea phase, but it is still a rough estimate. It does not include a detailed planning, more like a rough cut Gantt-chart like estimated planning that show milestones. This phase may take up to months to complete.

The next stage-gate phase is the initiation of the project (3), with a project initiation document as main deliverable. Again, information is more certain in this stage. Now, project managers have to request capacity at the capacity management office for the resources as described in the project proposal. All others requiring any capacity are submitting their estimated resource requirements to before a known deadline. Each quarter, capacity management office compares the demand for each resource to the availability of each resource. The demand always greatly exceeds capacity. Next, capacity is allocated to certain projects, while others will have to be postponed. Now, the project has dedicated resources allocated to them that they can use in certain time windows.

The subsequent stage-gate phase is the realisation (4a) and implementation (4b) of the project. Up to now, most effort was spend by the project management office in a supporting role. Now, mainly resources of operational departments and sections are required. This stage-gate phase is separated, since realisation is the development of a product or service, whereas implementation is making an order operational which can only happen after the realisation phase is completed.

Finally, a project is closed (5) and an evaluation follows to review if original goals are achieved and to close the budget. Regularly, a delivered order requires some sort of continuous effort such as upkeep or maintenance of some sort. That required effort is part of run management (see section 1.1).

To track the performance, a number of KPIs are periodically measured. Among others are 1) the number of orders completed in time, and 2) the number of projects that are not behind on schedule.

3.4 Analysis results

Unfortunately, the conclusion of the analysis is that the data at JIVC is insufficient to construct a PSP instance (subsection 3.4.1). Still, MinDef has ample reason to continue the development of a solution method (subsection 3.4.2). Therefore, a substitute dataset is acquired from literature (subsection 3.4.3).

3.4.1 Lack of suitable data

The current processes of CM and PM do not contain suitable data to generate a PSP instance. The main problem is three-fold.

Resource availability

The availability of resources is not readily available in any software suite, as concluded by the analysis and confirmed by (decentralized) planners. In the current process, many inquiries are made in a rather manual and time-consuming fashion to attain the availability of resources, costing a considerable amount of time. Furthermore, new inquiries have to be made when disruptions occur, something that as already established is bound to happen regularly under current conditions.

Resource availability data must be available for scheduling. Multiple work arounds could be devised however, but all rely on estimations that downgrade the solution quality. For example, a rough availability estimation could be acquired, which may already be able to generate valuable information and support planners in decision making.

Time-independent workload estimation

During planning and scheduling, workload is estimated, which is an essential requirement for planning. In the current process, the workload estimation is not explicitly being saved as a numerical value in any data model. What actually happens is that planners inquire the availability of involved resources. Based on that availability, the estimated workload is planned and a start date and finish date are saved into the system, similar to a Gantt chart planning. It is unknown what the expected duration of a task was.

Suppose for example that a planner estimates a workload at 40 hours and just checked that the corresponding resource is available for 20 hours a week. The estimated finish date is set two weeks after the start date, which are both registered in the system, but the requirement of 20 hours is not. Now, the major disadvantage becomes apparent when for any reason whatsoever the planning has to be revised. The task cannot simply be shifted because it became time-dependent to that particular time-window. It is also valid for other time-windows with equal or more availability, but that information is also not present in any system. Moreover, the process is not reversible, meaning that the workload cannot reliably be derived given the start and finish dates.

This problem can hardly be worked around. Scheduling is futile without knowing (estimated) workloads.

Tasks and relations between tasks

For the majority of work, (precedence) relations between tasks and required resources are not available in the current data models. For some projects, relations are available between mayor stages of a project, such as stage-gates and milestones, similar to how Gantt charts are usually constructed. For the three aspects, this one is least important.

The aspect could be worked around by only considering the available relations and executing the rest in parallel (or cascade depending on the context). As such, imposed relations may not reflect actual relations but estimate them. This would still yield solutions to the problem but have a negative impact the solution quality.

3.4.2 Continuation

The conclusion is that scheduling based on operational data from JIVC is not possible within the time-window of the thesis due to the lack of suitable data. Nonetheless, MinDef requests for the development of a solution method for their PSP based on the following reasoning. First of all, a new software suite is being acquired which replaces an old one and opens new possibilities to register project data. Furthermore, JIVC is working on continuous improvement processes to improve efficiency of their operations, and sees future value in the development of a solution method. Moreover, JIVC will be completely reorganised starting in the latter part of this thesis, which opens (future) possibilities for a proof of principle or pilot project. Additionally, manual construction of an instance is still possible, which could lead to important insights for the most important cases and could serve as a proof of concept for further automation. This is sufficient motivation for the development of a solution method for their PSP.

3.4.3 Substitute dataset

A substitute dataset is required to provide problem isntances and test a solution method in the continuation of this thesis. The scientific community created benchmark datasets to compare the performance of methods that solve particular PSPs. These datasets could serve as a substitute for the unsuitable data of JIVC.

Single project datasets

For decades, researchers in project scheduling use datasets to benchmark the various approaches in solving PSPs. First, manual datasets were constructed and the first influential one was that of Patterson (1984). As computing power increased and heuristics improved, the need for more difficult instances arose. Throughout the years, researchers developed project generators to generate instances for benchmarking. As described by Browning and Yassine (2009), the most notable were DAGEN, ProGen, RanGen, RanGen2 and HierGen. Although not being a dataset, Deblaere et al. (2011) provided a notable educational project scheduling software tool named RESCON. One of the most important datasets was that of Kolisch and Sprecher (1996), who presented a dataset to benchmark RCPSP algorithms for both single- and multi-mode instances. All their instances were generated by the generator ProGen as described by Kolisch et al. (1995). A recent dataset is provided by Coelho and Vanhoucke (2020), who investigated what makes an RCPSP instance hard and create a dataset of difficult instances. A collection of datasets is being curated on https://www.projectmanagement.ugent.be/.

Multi-project datasets

A need arose for multi-project problem instances. The MPSPLib is a multi-project project scheduling problem dataset specifically developed for problems subject to asymmetric information distribution. Most scientific literature solving this decentralized problem uses this benchmark set. The dataset is introduced by Homberger et al. (2007) and subsequently used in a MAS by Homberger (2007). The dataset is constructed from combinations of instances of the aforementioned PSPLIB. It contains 140 instances of different problem sizes. Instances have either 2, 5, 10 or 20 competing projects. Projects each have either 30, 90 or 120 tasks, and all projects within an instance have the same

amount. Each instance has exactly four resources, and at least one of them is global. They include special cases in which all resources are global, indicated as AgentCopp (AC) cases. Later, Browning and Yassine (2010) constructed a dataset of 12.320 instances, where each instance has 3 projects and 20 tasks. Still to this day, researcher find new solutions using novel solution methods, which they upload to http://www.mpsplib.com/ as a benchmark.

Similar to the generation of disruption scenarios in subsection 2.1.1, generated problem instances are bounded based on the parameter settings of the generator. Eynde and Vanhoucke (2020) provide an extensive overview of benchmark datasets for the RCMPSP. They describe characteristics of instances and the ranges that these characteristics span. This is important when benchmarking a solution method, because it implies that a good result for one dataset does not guarantee a good result on another dataset if they cover other characteristic ranges. This would limit the generality of a solution method. Knowing this, Eynde and Vanhoucke (2020) propose a new dataset that covers a broad range for each characteristic. They created three datasets, the first containing 833 instances having 6 projects each, the second containing 1463 instances having 12 projects each, and the third containing 2.254 instances, having 24 projects each. All projects have 60 tasks and four global resources.

Selection

Any of the three datasets by Eynde and Vanhoucke (2020) can be regarded as superior to the other ones described. Still, the MPSPLib dataset is chosen for the continuation of this thesis, the main reason being its popularity in literature which allows comparison. The dataset by Eynde and Vanhoucke (2020) is too new and it can hardly be compared to existing solution methods. The critique by these researchers regarding narrow band characteristics and the corresponding implications are taken into account during verification and validation of the solution method. One of the smallest problem instances, , is chosen as an example throughout this thesis and presented in Appendix H.

3.5 Uncertainty

The goal is to account for the expected uncertainty such that a robust schedule is generated that is able absorb that uncertainty to some degree. Using Figure 2.1, the uncertainty relevant to scheduling has a predominant internal focus. It has properties as regularly seen in IT, new product development and engineering to order projects. All three internal uncertainty factors are present.

The first is organizational uncertainty, which regularly impacts the operation in the form of unexpected priority changes originating from higher authorities within the organisation. These changes effect complete projects such that. Changes in project priority changes the order of whole projects, therefore often having so much impact that repairing a schedule is ineffective. Protecting against this uncertainty would require too much slack and is therefore out of scope.

The second is work content uncertainty, which is split up in four subcategories. Of those, task duration uncertainty is by far most common. Furthermore, requirement changes uncertainty is common, occurring in the planning phases of both CM and PM, where

work content can still vary. It effects operational project scheduling to a lesser degree, and mainly effects tactical capacity planning. Management estimates an increases in activity duration of 10 percent of an unspecified number of random tasks as realistic. This is based on an actual uncertainty expectation measure used in current PRINCE2 project management at JIVC. Since 10% uncertainty is expected, it is also chosen to relinquish at most 10% of the makespan in favour of robustness.

The third is resource availability uncertainty, which is also common at JIVC. Management estimates a 10% deviation of resource availability on a daily basis. Reasons include absence and reduced capacity due to uncertainty at RM requiring more resources at a higher priority. Management estimates a drop in resource availability between 0 and 10 percent for every day for every resource as realistic. This uncertainty is seen as less important compared to work content uncertainty. Therefore, this type of uncertainty is seen as part of future work.

3.6 Agent modeling

This section answers ***RQ2*** by representing the entities at JIVC as agents. Regarding every entity at JIVC as an agent is inefficient. As indicated in the previous chapter, a market type is an efficient structure to allocate resources (Horling & Lesser, 2004). The current capacity management board already uses the term 'demand-supply coordination' for their quarterly plans. Therefore, a demand-supply perspective that corresponds to the 'market' organizational type is considered to aggregate entities. The entities at the MinDef should be represented by agents. The three classic agent types are used as a representation, being demand (subsection 3.6.1), supply (subsection 3.6.2) and mediation (subsection 3.6.3).

3.6.1 Demand

The demand side represents the first set of aggregated entities (Figure 3.2).

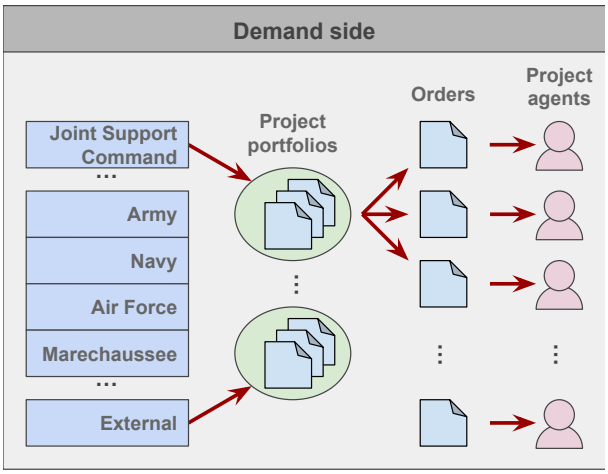


Figure 3.2: Agent representation of demand side

Clients (branches of defence) have orders that demand resources from JIVC for their completion. All branches have a portfolio of projects. These branches are subjected to their own constraints (budget) and the constraints of JIVC (capacity) and can therefore

not execute all orders. In most organisations, the completion of a project is associated with a monetary value, at the MinDef this is a strategic value. The branches therefore want to choose projects such that the total strategic value is maximized. This is a decision at a strategic level which is out of scope.

The chosen projects become orders at JIVC. The orders pass through a number of entities that are all momentarily responsible for processing before it arrives at an entity responsible for execution of that order. Each entity is inclined to finish their own project within their assigned due date. Moreover, they aim to minimize the makespan or tardiness. Alternatively, they could aim to minimize resource costs to get the most out of a limited budget. This entity is ideally represented as a project agent in the MAS.

3.6.2 Supply

The supply side represents the second set of aggregated entities (Figure 3.3).

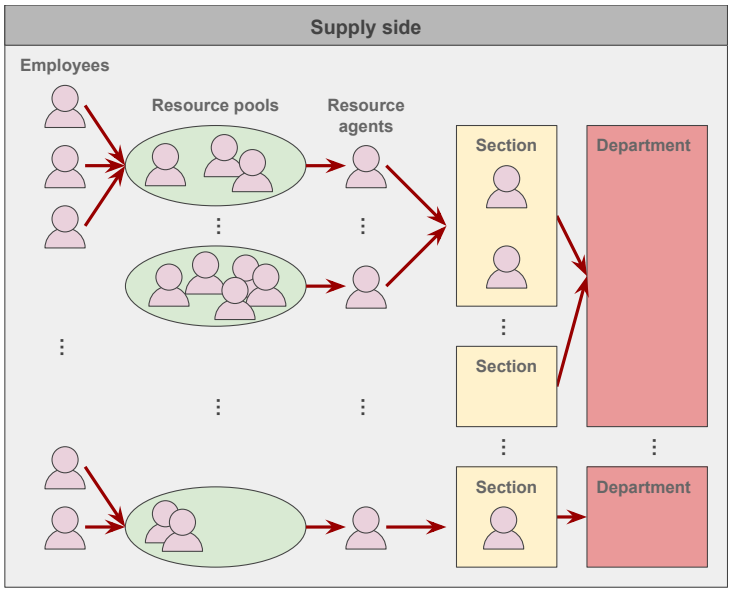


Figure 3.3: Agent representation of supply side

Employees supply demand by spending their limited time on work such as orders. They are part of resource pools (section, e-CF-profile, specialization) that represent their aggregated capacity. This entity is ideally represented as a resource agent in the MAS. They aim to maximize utilization or maximize resource leveling.

All resource pools belong to a section. It is alternatively possible to represent sections as resource agents that control the resource pools within a section. The same applies to departments that are one level higher in the hierarchy. Objectives may change if resource agents represent entities higher in the hierarchy. At that level, DMUs may minimize the costs or maximize 'revenue'.

Resource pools rarely run out of work since demand greatly exceeds supply. Therefore, management at JIVC argues that resource leveling currently has currently less priority. Yet, preventing resource pools from being utilized too much can be considered part of resource leveling. High utilization levels are relevant, since it causes a loss of flexibility in responding to uncertainty, hence decreasing throughput. The cause and effect diagram

identifies this as the feedback loop of disruptions (see subsection 1.2.2). Therefore, resource pools should be protected from high utilization rates. Besides resource leveling, this can also be achieved via robust scheduling methods. These generally add slack to schedules, and therefore some degree of flexibility to absorb uncertainty.

3.6.3 Mediation

Mediation is required for coordination and to resolve conflicts that arise between the demand-side and supply-side (Figure 3.4).

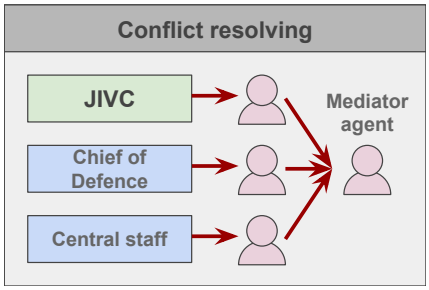


Figure 3.4: Agent representation for conflict resolving

Decisions have to be made to resolve conflicts. Direction in decision making is provided by a long term strategy, and that strategy is determined at the highest level in the organizational hierarchy. There are three relevant entities. The first is the chief of defence which resolves conflicts between operational branches and the second is the central staff which resolves conflicts between the other branches. The third agent consists of a number of entities at JIVC comprised of overarching management and overseeing boards throughout its hierarchical layers. The mediator agent is therefore represented as these three entities as they strive to resolve conflicts by looking which decision align best with strategic goals. The agent strives to maximize social welfare or maximize fairness.

3.7 Conclusion RQ1 and RQ2

To conclude this chapter, a summary is given of the answers to both research questions.

RQ1: Data can not be acquired at JIVC to create PSP instances. The MPSPLib dataset from literature is used as a substitute for problem instances due to the lack of suitable data. The ***RQ5*** should support this structure. There are two kinds of uncertainties encountered that should be accounted for in scheduling, being a 10% increase in duration of tasks and a 10% drop in resource availability for every day for every resource. The robust project scheduling methods to be selected in should protect against this uncertainty. The RDPSM designed in ***RQ6*** should be subjected to this uncertainty. This concludes the first step as described by Billaut et al. (2008).

RQ2: Three categories are introduced that represent a 'market' structure, which is efficient for resource allocation. Entities at the MinDef are appointed based on those having a demand for resources, those supplying resources and those coordinating and mediating interactions. All entities within one of these categories are by either project

agents, resource agents or a mediator agent. The most common aims (or objectives) of each agent type is indicated. A MAS in ***RQ5*** should support this structure.

Chapter 4

Literature study

The previous chapter investigates the problem at JIVC. Three research questions can be answered now the situation at JIVC is understood. This chapter closes three knowledge gaps by consulting literature.

RQ3 is answered by reviewing literature for suitable measurements for the quality of a solution and the performance of a project scheduling method (section 4.1). ***RQ4*** is answered by reviewing literature for suitable robust project scheduling methods that deal with the uncertainty as identified in ***RQ1*** (section 4.2). For both research questions, surveys and publications are used from recognized and established sources within the field of project scheduling. ***RQ5*** is answered by conducting an SLR to identify the most suitable multi-agent system that is truly decentralized and deals with problem type as identified in ***RQ1*** (section 4.3). This chapter is concluded with a summary of the answers to the research questions (section 4.4).

4.1 Measures

This section answers ***RQ3***, which consists of two parts. The first part focuses on measuring the quality of a single solution (subsection 4.1.1), and the second on the performance of the RDPSM itself when the solutions are subjected to the uncertainty of JIVC to evaluate the treatment in ***RQ7*** (subsection 4.1.2).

4.1.1 Schedule performance

The business problem states two aspects of interest, being 1) reduced in-time delivery (unreliability) and 2) longer makespans. Therefore, a measurement is required to assess the reliability and makespan. Measuring the makespan is self-evident and for completeness described in Appendix A. For increasing reliability, robustness must be evaluated, taking into account the type of uncertainty at JIVC as identified in ***RQ1***.

First, the term 'surrogate measure' is introduced as explained by E. Demeulemeester and Herroelen (2010). Naturally, a measure for robustness is supposed to indicate the robustness of a schedule. An important problem is that determining the expected robustness is unrealistic due to its computational hardness. Therefore, robustness is generally determined by means of a simulation that subjects a schedule to uncertainty, which

approximates robustness. Simulations however are also relatively computational intensive. Researchers developed surrogate measures to solve this issue. These measures estimate the expected robustness of a schedule, which is less reliable as an exact computation or simulation, but can be calculated quickly and efficiently.

Researchers agree that robustness is not a single clearly defined concept, as for example indicated by Billaut et al. (2008) and Rasconi et al. (2008). Therefore, robustness cannot be defined as a single measurement. One of the most recognized views on robustness splits robustness in two categories as proposed by W. Herroelen and Leus (2005). These are quality robustness and stability robustness. The aim is to identify and select one or more suitable measurements to quantify the solution quality.

Quality robustness

Quality robustness is the concept that considers how a performance measure (or objective function) of a schedule differs before and after it was subject to uncertainty. For example, given the most common objective function of the makespan, the makespan of the schedule is compared to the actual makespan after execution. Other quality examples are earliness-tardiness and those monetary based such as cost or Net Present Value (NPV). The less a performance measure degrades, the better the quality. The majority of robustness measures are based on slack, because shifting tasks in a time-window that does not effect other tasks and therefore preserves quality.

Literature is reviewed to explore measures. Hazır et al. (2010) introduced surrogate metrics suitable for the RCPSP, which Hazır et al. (2015) later classified in six main groups as described in Hazır and Ulusoy (2020). These are 1) average slack, 2) weighted slack, 3) slack utility function, 4) dispersion of slacks, 5) percentage of potentially critical activities and 6) project buffer size. Having replicated several project settings and calculating the correlation, Hazır et al. (2010) found project buffer size and weighted slack, with weights equal to the number of immediate successors of the activities to be the best predictors of robustness.

Khemakhem and Chtourou (2013) reviewed robust measures found in literature and did an extensive experiment on their performance. Additionally they devised robustness measures themselves. The main uncertainty factor is an increase in activity duration.

Khemakhem and Chtourou (2013) introduce new concept, being 'sufficient slack'. A task has 'sufficient slack' if it has enough slack to absorb a certain expected disruption. This expected disruption is expressed as a percentage, for example 20%. Suppose a task duration of 10, in that case, a task has sufficient slack if it can absorb a duration of $10 \cdot 20\% = 2$. The sufficient slack could be a non-integer number such as a fraction, for example if the duration was 9 instead of 10. Particular attention should be given since a duration is always integer. Although not specified in the research, it seems intuitive to apply a ceiling function in stead of rounding, such that the duration is always rounded up and the percentage is always contained in the duration.

The result of the experiments of Khemakhem and Chtourou (2013) result in the identification of the three best performing measures in terms of having the best correlation between objective function and robustness. The highest correlating measure is the Combined Slack Sufficiency (CSS) (Equation D.21). For each task it calculates a score. A score of 1 is given to a task if its slack equal to or larger than the 'sufficient slack', which means

that it is capable of absorbing that uncertainty. An additional score of 1 is given to each anterior task if the task at hand has a slack greater or equal to the 'sufficient slack' of that anterior task. An anterior task are all tasks This is calculated for all tasks of a schedule, aggregating a total score. The score is not on a scale and can therefore not directly be compared to other project instances. A user only knows that a higher value has a higher probability of being able to absorb uncertainty without degrading quality. Essentially, the performance measures indicates that a project can gain most robustness if a task has enough slack to absorb a certain percent of increase in task duration and enough slack to absorb those of anterior tasks. This makes intuitive sense, an early perturbation has a greater effect (snowball effect) on all succeeding tasks.

Stability robustness

Stability robustness is the concept that considers how much of a schedule is affected by a disruption. It evaluates the (in)sensitivity of a project after disruptions occur. High quality stability is a desirable and highly sought after property, since it indicates that even though disruptions occur, the original schedule can be executed as scheduled to a high degree. Unstable, sensitive or nervous schedules have to be rescheduled more often. This is undesirable in most practical situations, because in real life rescheduling is often accompanied with costs through additional work (in administration or communication for example).

The most used stability measure is one that sums the absolute deviation between scheduled start times and actual start times, for example as provided by the 'sensitivity' measure of Rasconi et al. (2008) (Equation D.22). Additionally, some add a weight to each deviation that resembles disruption costs. Other stability measures may be sum of disrupted tasks or the sum of rescheduled tasks.

There are some scheduling policies that effect stability robustness. Deblaere et al. (2007) indicate that some researchers advocate for so called 'railway executions'. In this mode, tasks cannot be started before their starting times as scheduled in the baseline schedule, making them similar to job shop release dates. This increases stability. Other researchers advocate the 'roadrunner mentality', which is the opposite of railway scheduling. In it, tasks are started as soon as possible, expecting that it decreases disruptions in succeeding tasks due to increased slack. An example of stable project schedule generation is that of W. Herroelen and Leus (2004).

Quality versus stability

Quality robustness and stability robustness are separated concepts. A disrupted schedule may attain quality (for example have no delay in makespan) while all tasks had to be rescheduled to other starting times and hence losing all stability. On the other hand, a schedule may have preserved most stability (most tasks executed at scheduled times) but still miserably fail in its quality measure due to one unfortunate uncertainty.

4.1.2 Method performance

The previous sections explored measures focused on single project instances. Other measures are required to assess the performance of a method, in this case the RDPSM. A

common measure for all computational methods is the evaluation of the computational time by measuring the difference between the end and start times of the method.

Additionally, two common measurements are 1) the percentage of scenarios in which the objective value did not degrade after disruptions, and 2) what the average difference is of the objective value after disruption. Suppose the objective of a makespan, it would indicate how many percent of the scenario's are completed within the original makespan and what the average increase in makespan is of all scenarios.

4.1.3 Selection

For JIVC, communicating reliable due dates is key. Reliability is achieved if and only if a schedule subjected to uncertainty is finished within its expected due date. Quality robustness is the measure that can quantify the in-time delivery or 'reliability' as meant in the problem statement of JIVC, hence it is selected to be used in the thesis in favour of stability robustness. The CSS measure by Khemakhem and Chtourou (2013) has the highest correlation with quality robustness of a schedule, therefore it is chosen to be used. To evaluate the performance of the method, % of instances that are completed within their indicated makespan after being subjected to uncertainty is used.

4.2 Robust scheduling methods

This section answers research question ***RQ4*** by identifying the most suitable robust scheduling methods to deal with uncertainty in the PSP at JIVC. The literature on robust project scheduling is vast, therefore, the scope is narrowed down by filtering. First, four out of five types of robust scheduling methods presented in the preliminary knowledge (Figure 2.2) are excluded based on the situation analysed in ***RQ1***. Next, within that selection, two publications are used that guide researchers and practitioners in selecting appropriate methods, and those are presented (subsection 4.2.2).

4.2.1 Type selection

In the preliminary knowledge (subsection 2.1.2), These are investigated to determine which types could yield relevant methods to solve the PSP at hand.

First, two types are ruled out as candidates. Type 5, sensitivity analysis, is ruled out because it contains procedures that do not generate but measure the ability to sustain uncertainty. This is less relevant since the goal is to explore methods that generate schedules. Type 3, stochastic project scheduling, is also ruled out. Probability distributions and sorts are required for stochastic scheduling, but projects and tasks at JIVC have such a unique nature that meaningful probability distributions cannot be obtained. Moreover, stochastic methods assume independent variables, which is often violated in scheduling because tasks depend on each other. Three types are left over, which are explored in more detail.

Type 4, fuzzy project scheduling, may be interesting since it deals with vague information. Additional information on fuzzy scheduling is required in order to determine if it suitable to use for the situation of JIVC. This information is presented in Appendix F. In conclusion, most fuzzy scheduling methods require multiple estimates for the duration of a single

task (such as an expected, best case, and worst case scenario duration). This additional information is not available. Furthermore, it is unlikely that employees are eager to enter additional information for every task of every project in a future situation. Simply estimating or generating that additional information is also not suitable, for the same reason that stochastic scheduling is ruled out. Therefore, fuzzy project scheduling is ruled out.

Type 2, reactive project scheduling, is the most promising approach to deal with PSPs subject to a high degree of uncertainty. As indicated in Figure 2.3, there are multiple degrees to which a reactive component could be integrated in a scheduling method. The most extreme, a completely online or dynamic method does not even generate a schedule but merely selects the most suited task in real-time for each resource given the available workload at that moment. First of all, components with the highest degree of online scheduling are ruled out. As indicated, these only have a short and limited time-horizon, whereas JIVC wants to generate stable schedules which they can act upon. Unfortunately, other reactive scheduling methods are also excluded. All reactive scheduling methods require some kind of (near) real-time feedback at moments that disruptions occur. At this moment, the scheduling process and IT-infrastructure is not suited to provide this kind of feedback. It would require that employees register completed tasks as soon as these are completed and register disruptions as soon as they occur. This is infeasible at this moment and in the near future. Hybrid models (pro-active and reactive) are therefore also excluded.

Type 1, pro-active project scheduling, is the only remaining type and is a feasible type as it is generated up front using a known PSP instance. Therefore, literature is reviewed regarding robust scheduling methods.

4.2.2 Pro-active project scheduling methods

W. Herroelen and Leus (2007) and Hans et al. (2007) assist researchers and practitioners by providing a general direction to robust scheduling methods that are most likely to work in certain situations. A situation is categorized based on the dependency and variability (Table 4.1).

Table 4.1: Different approaches to the (multi-)project scheduling problem, from W. Herroelen and Leus (2007)

	totally dependent	*rather dependent*	*rather independent*	*totally independent*
low variability	7. stable plan; satisficing	5. stable drum; efficient remainder	3. efficient drum; efficient remainder	1. deterministic
high variability	8. process mgmt.; rough plan with sufficient slack	6. stable plan with queuing	4. stable drum; dispatch or predictive-reactive	2. dispatch or predictive-reactive

At JIVC, there is a high variability due to the uniqueness of projects and the accompanying uncertainty. Furthermore, the situation is rather dependent (case 6) or totally dependent

(case 8) due to all of the shared resources between projects. W. Herroelen and Leus (2007) write regarding these two cases: *"tight milestones will induce a permanent 'fire-fighting' mode, since lead times are hard to estimate and strongly depend on the current organizational load."*. This is exactly what JIVC encounters, confirming the observations and categorizations made previously in this thesis. W. Herroelen and Leus (2007) advice a strategy using "a robust drum plan" without too much detail in order to respond to uncertainties. The scope is narrowed down to these methods.

In the book on project scheduling by E. Demeulemeester and Herroelen (2010), an overview is presented of project scheduling methods (Appendix E). Based on the conclusion in the previous section, the direction given trough W. Herroelen and Leus (2007), the focus is on proactive methods. The book presents a range of robust project scheduling methods, four of which seem suitable (Table 4.2). The description of these candidate methods concludes this chapter, as the most suitable method depends on the core MAS framework of the following section.

Table 4.2: Candidate pro-active project scheduling methods selected from E. Demeulemeester and Herroelen (2010)

Method	**Method type**
Critical Chain Method	Time buffer insertion
Starting Time Criticality	Time buffer insertion
Partial Order Schedules & chaining	Resource flow network
Myopic Activity-Based Optimization	Resource flow network

Time buffer insertion

Time buffering is the addition of time buffers in a schedule. One of the most popular methods is the Critical Chain Method (CCM) surrounding the Theory Of Constraints (TOC) as described by Goldratt (1997). CCM inserts particular buffers between certain tasks, therefore CCM is also known as buffer management. Using regular project scheduling methods, the Critical Chain (CC) is identified similar to how the CP is identified.

In single project scheduling, three types of buffers are added which attempt to protect the CC from being delayed. A project buffer is added at end of the project to protect the customer due date. Additionally, feeding buffers are added at non-CC tasks, such that a delay in those chains will not affect the CC. Finally, resource buffers are added in front of CC tasks when that resource starts working on CC tasks to account for delays of the resource in non-CC tasks. In multi-project scheduling, projects are prioritized and the most constraining resource is identified as the drum resource. Everything is scheduled around the critical drum resource, and drum buffers are added to protect that.

Buffer sizes have to be estimated, and the two common methods to do that are the 50% rule and the Root Square Error Method (RSEM). The latter generally performs better but requires more estimates. CCM is more generally more focused towards achieving quality robustness instead of solution robustness. Ghaffari and Emsley (2015) produced an overview of CCM as a project management paradigm. The performance of buffers was investigated by Van De Vonder et al. (2005).

The Starting Time Criticality (STC) procedure is also a method that inserts buffers (Van de Vonder et al. (2008)). Given a schedule without any buffers, the method assesses the

criticality of the starting times of tasks, named the STC. the criticality is the probability that a task cannot start as scheduled. Subsequently, it adds a safety buffer to the task that improves stability robustness best in iterations, until no improvement is possible. The probability is very difficult to determine, therefore an approximation is used. According to E. Demeulemeester and Herroelen (2010), this fairly simple method was at the time of writing one of the best.

Resource Flow Network

Resource flow networks, as for example introduced by Artigues and Roubellat (2000), regard the problem as a transshipment network. They mainly focus to attain stability robustness by focusing on resource allocation in baseline schedules, and an instance is named a resource flow network. Because it is seen as from a network perspective, each network contains a range of schedules composed of the feasible start and end times of tasks. According to Deblaere et al. (2007), a resource flow network has a serious impact on robustness. Klimek and Łebkowski (2011) define criteria to evaluate resource flow networks.

A particular resource flow network strategy is proposed by Policella (2005), being Partial Order Schedule (POS) in chaining form. A scheduling problem is considered as a graph with their usual precedence constraints and additionally, temporal and resource constraints are added onto that graph as edges. In a POS, tasks are not fixed at specific starting times, but are allocated depending on the available interval in the temporal graph. Therefore, a POS is a set of solutions that can be represented by such graph, thereby containing many normal 'schedules' as solutions. Policella et al. (2008) apply chaining to the POS, first a schedule solution is generated, after which it is turned into a graph using a heuristic named 'chaining'. The chaining procedure returns a POS, thus a set of solution, from an initial schedule, one of which could be chosen for execution or for repairing. Another strategy using POS is the three step Myopic Activity-Based Optimization (MABO) procedure as presented by Deblaere et al. (2007).

4.3 Decentralized project scheduling

This section answers ***RQ5*** by identifying the most suitable multi-agent system for the situation at JIVC. Not all method are truly decentralized, but as established in section 2.2, those that employ protocols such as negotiations or CAs are most often truly decentralized. Therefore, an SLR with a narrow scope is conducting for this purpose, presenting the search query (subsection 4.3.1), the search results (subsection 4.3.2) and a summary of the publications (subsection G.0.1). Finally, the most suited method to employ as core framework is selected (subsection 4.3.3).

4.3.1 Search query

The search term query is (*"project scheduling"*) **AND** (*"negotiation"* **OR** *"combinatorial auction"*)) in the title of the article. The search results are limited by only accepting papers including the keywords in the title. This (very) restrictive search is applied due to the preliminary research in Martens (2021). The language is set to English and there is no publication date range. The online sources are 'ACM Digital Library', 'IEEE

Xplore', 'ProQuest', 'ScienceDirect', 'Scopus', 'SpringerLink', 'Wiley Online Library', and 'WorldCat'. All unique occurrences are filtered, and the following exclusion rules apply: 1] the topic is a CA or negotiation regarding a DRCMPSP, and 2] the publication is available and accessible online in the English language. The search query results are presented in Table 4.3.

Table 4.3: Query results

Source	**Publications**
Scopus	15
WorldCat	12
IEEE Xplore	2
ScienceDirect	2
ProQuest	1
ACM Digital Library	1
SpringerLink	0
Wiley Online Library	0
Total unique results	**16**
Total unique results	15
Excluded by exclusion rule	7
Final result	**8**

4.3.2 Search results

The publications in the final search results are presented (Table 4.4). On closer inspection of the papers, it was discovered that two pairs of papers are almost equal but not exactly the same, and therefore counted as unique occurrences. Because these are closely related, they are considered together. All eight publications are summarized (Appendix G).

Table 4.4: SLR combinatorial auctions and negotiations in project scheduling results

#	DOI	Citation	Title
1	10.1080/00207540500066796	Lau et al. (2005)	Distributed project scheduling with information sharing in supply chains: Part i - An agent-based negotiation model
2	10.1109/ICMLC.2009.5212288	Li (2009)	Mas-based negotiation mechanism for ship multi-project scheduling
3	10.1016/j.engappai.2011.12.003	Adhau et al. (2012)	A multi-agent system for distributed multi-project scheduling: An auction-based negotiation approach
4a	#N/A (Conference proceedings)	Song et al. (2016)	Decentralized multi-project scheduling via multi-unit combinatorial auction
4b	10.1007/s10458-017-9370-z	Song et al. (2017)	A multi-unit combinatorial auction based approach for decentralized multi-project scheduling
5a	10.1109/CYBConf.2017.7985794	Cheng and Lo (2017)	Multi-project scheduling by fuzzy combinatorial auction
5b	10.6186/IJIMS.201906_30(2).0004	Cheng et al. (2019)	Solving multi-mode resource-constrained multi-project scheduling problem with combinatorial auction mechanisms
6	10.1016/j.ejor.2017.03.022	Homberger and Fink (2017)	Generic negotiation mechanisms with side payments – Design, analysis and application for decentralized resource-constrained multi-project scheduling problems

4.3.3 Selection

The eight publications are rated on five aspects (Table 4.5). The first two are requirements that are either met or not. The first requirement #1 is a deterministic solution method such that a unique input and parameter configuration always results in the same output. The second requirement #2 is project scheduling method suitable for the problem type at JIVC. The remaining three aspects are ordered from most important to least important, and each aspect is rated on a scale of four units (–, -, +, ++). The third requirement #3 is the performance of the method in solving the problem. Comparing performance is difficult due to differences in objectives, validation and verification. Performance is seen as the combination of computational efficiency and degree in which the objective is achieved (such as average project delay or total makespan). A positive score means that it performs better in comparison to other methods. The fourth aspect #4 is the problem instance input. A positive score is allocated if it is compatible with the MPSPLib problem instances, whereas a negative is allocated whenever this is not possible. The fifth aspect #5 is the agent types. A positive score is allocated if the method is compatible with the agents identified in section 3.6 whereas a negative score is allocated when this is not the case.

Table 4.5: Selection of publication from SLR results

Citation	**#1**	**#2**	**#3**	**#4**	**#5**
Lau et al. (2005)	✓	✓	-	-	+
Li (2009)	×	✓	–	–	++
Adhau et al. (2012)	✓	✓	-	++	+
Song et al. (2016)	✓	✓	+	++	-
Song et al. (2017)	×	✓	+	++	-
Cheng and Lo (2017)	✓	×	+	+	-
Cheng et al. (2019)	✓	×	+	+	-
Homberger and Fink (2017)	×	✓	++	+	-

Four publications do not meet the first two requirements. The publication of Li (2009), Homberger and Fink (2017) and Song et al. (2017) do not provide a deterministic solution method. Both Cheng and Lo (2017) and Cheng et al. (2019) are focused towards the multi-mode scheduling problem, a variation that is not applicable for the problem at JIVC.

The publications of Song et al. (2016) meets all requirements and scores highest on the remaining aspects. The publications indicates that it outperforms that of Adhau et al. (2012) and it uses the MPSPLib benchmarking dataset indicating equal input. This publication does not feature resource agents that use private information which make it less appealing, however, their objective could be realised by a robust method as indicated in subsection 3.6.2.

4.4 Conclusion RQ3, RQ4 and RQ5

To conclude this chapter, a summary is given of the answers to both research questions.

RQ3: For analysing the robustness of a schedule, quality robustness is most suited because it represents the in-time reliability as stated in the problem statement of JIVC. The CSS measure by Khemakhem and Chtourou (2013) is chosen to grade robustness of a schedule since it has the highest correlation of quality robustness according to their research. The % of cases that a schedule is completed within its makespan after it is being subjected to uncertainty is used to measure the performance of the solution method.

RQ4: Various PSP methods are explored that generate robust schedules. Using literature, the range of possible robustness methods is reduced to a small subset of suitable measures. ***RQ6*** requires one or more methods that generates robust solutions in some way, and the methods presented in this section could be used for that.

RQ5: An SLR is conducted that identified eight publications that are truly decentralized project scheduling methods that employ either a negotiation or CA. The method by Song et al. (2016) is identified as most suitable and serves as the core framework in the RDPSM.

Chapter 5

Treatment design

The previous chapter closes three knowledge gaps and identifies the most suitable decentralized project scheduling method and several suitable robust project scheduling methods. This chapter answers ***RQ6*** by designing a RDPSM. To accomplish this, the MAS of Song et al. (2016) identified in ***RQ5*** is used as a core framework which is extended with robust methods identified in ***RQ4***.

First, the method of Song et al. (2016) is explained (section 5.1). Next, the replication process is presented (section 5.2). Subsequently, a suitable way to implement robustness is devised based on the methods as identified in ***RQ4*** (section 5.3). Next, modifications made to the original method are presented (section 5.4), the capabilities of the RDPSM are presented (section 5.5) followed by identified improvements (section 5.6). Finally, the chapter is concluded by answering ***RQ6*** (section 5.7).

5.1 Main solution method

The method of Song et al. (2016) is chosen as the main solution method. This method consists of project agents that participate in an auction led by an auctioneer. First, a project instance is introduced (subsection 5.1.1). Next presented are the agent set-up (subsection 5.1.2), bidding process (subsection 5.1.3) and the auction rounds (subsection 5.1.4).

5.1.1 Instance

An instance is equal to an MPSPLib instance combined with some additional information. An instance is introduced starting with the sets and subsequently the decision variables and input variables. A mathematical description of the sets, decision variables and input variables of an problem instance and an example of an instance are presented in Appendix H. A brief summary follows.

A problem instance includes seven sets. The first two sets are a set of time epochs on the time horizon and a set of global resources. The third is a set of projects. Each project in that set has a set of local resources and a set of tasks, and each task has a set of predecessors and successors, which are both subsets of the set of tasks.

The decision variables are starting times of tasks, which is an epoch in the set of the time epochs. All starting times of a project is a vector called a schedule, and all schedules is called a solution to the problem.

There are nine input variables. First, there is a time horizon. Next, there is a local and global resource level for each resource on each time epoch. Subsequently, each task has a duration and a resource requirement for each global and local resource. Each project has an earliest start date, revenue at completion and unit delay penalty.

5.1.2 Agent set-up

A high level overview of agent interaction is presented (Figure 5.1).

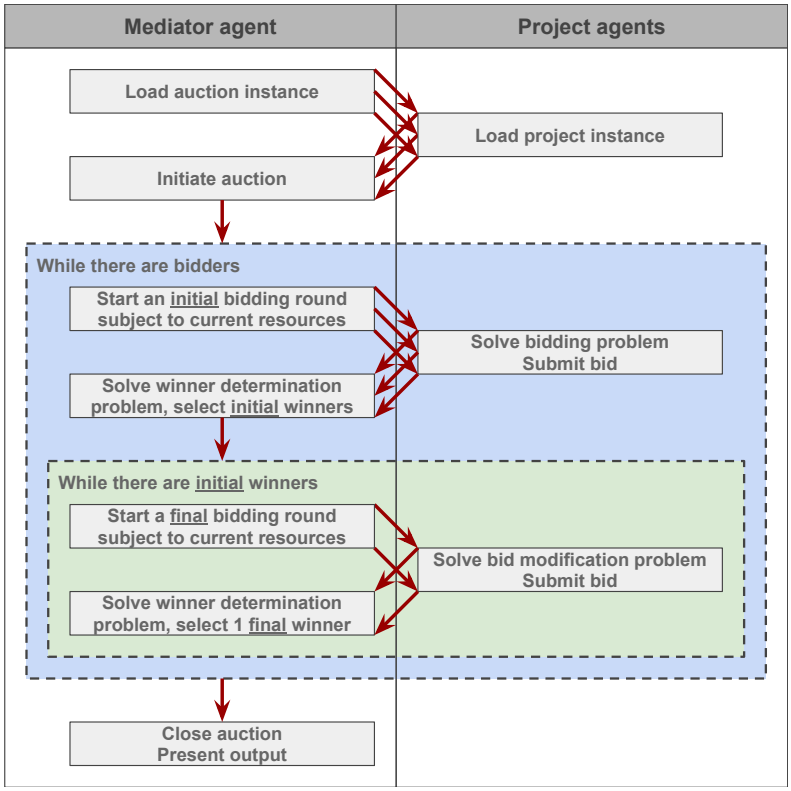


Figure 5.1: Abstracted agent interaction of method by Song et al. (2016)

The mediator agent (or auctioneer) prepares the auction by loading the project instances. The multiple red arrows in the figure indicate that a 'command' is send to all project agents to perform an action. These actions are independent of other project agents and can therefore be executed in parallel. All agents reply and the project agent is executing a while-loop for as long as there are active bidders that did not yet win in the auction.

Each project agent manages 1) a single project of which the contents are private information, and 2) a number of local resources that are only relevant to that project. The goods in the auction are global resources available on each integer epoch over a certain integer time horizon. The auctioneer manages all goods, meaning that this setup does not include explicit resource agents, the auctioneer takes upon himself this task. The auction is organized in multiple rounds and continues until the auctioneer allocated resources to all the project agents.

5.1.3 Bidding

Project agents are allowed to bid. A bid is a tuple generated by a project agent that consists of a multiset and a valuation of that multiset. A multiset represents the total global resource requirement of the agent. This is represented as a matrix, having the size of the time horizon in one direction and the number of global resources in the other. The resource requirement is determined by generating a schedule given the current global resource availability. A parallel SGS using the MINLFT priority rule is employed to generate schedules. In the publication of Song et al. (2016), the multiset corresponding to the generated schedule is called a core. The valuation of the multiset is the revenue of a project minus the incurred delay costs. The multiset only represents the resource requirement, it does not include any information on tasks, their start times or durations. Therefore, private information is conserved through bidding.

5.1.4 Auction rounds

Each round consists of two stages. In the first stage, project agents submit bids and the auctioneer selects provisional winners. In the second stage, provisional winners are allowed to modify their bids. Now the auctioneer selects final winners one by one until the auctioneer allocated resources to all provisional winners. This is repeated until all agents are processed.

Phase 1: initial bid submission

In a round, all project agents who have not yet been allocated resources submit a bid. The auctioneer ranks all bids based on a bid dependent sorting criteria. First of all, the auctioneer calculates the element-wise division of the resource usage by the available resources. The sum of that number is calculated by summarizing over both dimensions. The sorting criteria is the valuation of the bid divided by the square root of that sum. Starting at the highest ranking bid, resources are **provisionally** allocated to each bidder of which the bid does not intervene with any of the previous bids. This maximizes social welfare. Bids intervene when the total resource capacity is exceeded.

Phase 2: bid modifications

All provisional winners are now given the change to modify their bids. The goal is to move tasks with free slack (and can therefore be shifted in time) from high-demand time slots to low-demand time slots, hence creating room for other projects to be scheduled at those slots.

First, the auctioneer calculates the demand ratio, which sums the multisets of all bids while preserving the other matrix dimensions. The demand ratio directly indicates which goods are in high demand. The demand ratio is calculated to all bidders.

Next, all bidders modify their bids as follows. A project agents determines the set of tasks that have free slack and can therefore be shifted in time. Moreover, a 'resource index' is calculated for all shifts of all tasks in the set. This is the element-wise multiplication of the multiset and the demand ratio after which the result is summarized over both dimensions into a single number. A schedule using resources in high demand results in a large multiplication and hence a large number. Therefore, the project agent searches for

the shift in tasks resulting in the lowest resource index. The project agent repeats this until all tasks in the set are shifted.

Finally, the auctioneer determines final winner(s) using the same method as before. All losing projects modify their bid again and repeat phase 2 until all project agents have been allocated resources.

5.2 Replication

First, the replication process is presented (subsection 5.2.1). Next, the shortcoming of the replication is explained (subsection 5.2.2).

5.2.1 Process

The publication of Song et al. (2016) is recreated in Python, version 3.9.7, having the following (most relevant) dependencies: IPython 7.29.0; numpy 1.21.2; and matplotlib 3.4.3. Although efficient coding practices are applied where possible, the code is not specifically optimized for efficiency. For example, in a bidding phase, each bid could be efficiently generated in parallel but parallel computation or multi-threading are not implemented.

Parsers

Two parsers are programmed to import the data into Python and process it. One parser imports .xml-file MPSPLib instances and one imports the .bas-file Kolisch RCPSP instances, which is interpretable as a .txt-file. A full instance example is presented in Appendix H.

Assumptions

A few assumptions are made in the replication. The publication does not report on all parameter values that are used to generate the results that they report on, such as the revenue. Therefore, assumptions are made for these parameters. Furthermore, not all algorithms are explained in detail, meaning that some obscurity remains, for example in tie-breaking conditions in selection algorithms. In that case, simplicity is leading in resolving the issue, as for example tie-breaking based on the minimum task index number.

Modifications

Additionally, a few modifications are made that do not impact the workings but build towards a modular framework. The most meaningful is the introduction of a single resource agent where none existed in the original method. This resource agent owns all global resources and is the only one authorized to mutate capacity. The resource agent could be split up into multiple resource agents that control a single resource. The introduction of a resource agent has at this moment not a particular role and does not follow private information rules as it directly communicates all information towards the auctioneer. Though, the auctioneer is included such that objective functions can be given to it in future development.

Plotting

The decision variable of a PSP is a schedule with starting times for each activity. It is very attractive for practical reasons to visualize the resource usage over time. A vast majority of researcher and practitioners working with PSPs, even the most renowned ones, plot schedules using rectangles for tasks. As Csébfalvi (2012) proves, this is incorrect and may lead to incorrect plots. Moreover, when these rectangles are part of the solution method, it even leads to less optimal solutions. The correct way to plot visualizations is via a heightmap of the resource usage. This procedure is also adopted in plots for this thesis as shown in the output as described next.

Furthermore, it is interesting for DMUs from the supply side (such as sections and resource pools) to see the scheduled resource utilization over time. Therefore, the usage of each resource is plotted over time as a percentage of its total capacity at that epoch. This is an important managerial insight, as indicated by W. Herroelen and Leus (2007), utilizing a resources too much should be avoided because overloading results in a non-linear increase in throughput. Two lines are presented in each plot, a dotted line representing the value at each epoch and a solid line representing the average of three time units. Both can be important for general managerial decision making, since the average negates outliers to detected a high or low utilization over multiple time units whereas the other can be used to act upon the effects of outliers. The three day time unit is a variable that can be changed to any positive odd number except 1. Also, a green background band is drawn between the 60% and 80% utilization line to indicate a preferred utilization zone. These values are for illustrative purposes and are also variables to be selected by the user.

Output generation

Due to asymmetric information, the output is split in a private part for individual projects and a public part for the whole instance. As an output for individual projects, .txt files are generated for the project its characteristics and the project its measures, .pdf files are generated for the the critical schedule, the final schedule and a resource utilization graph over time. As an output for the whole problem, .txt files are generated for the problem its characteristics, the problem its measures and a complete log file of the auction, .pdf files are generated for the the anonimized resource usage schedule, a utilization graph over time, and all schedules of all the bids submitted in all rounds. Plotting schedules requires an enormous computational effort, therefore a settings file is provided to toggle every single output using Boolean values. The characteristics of the example instance mp_j30_a2_nr2 are presented in Appendix I, and the output in Appendix J.

5.2.2 Shortcomings

At this moment, the replication does not generate robust schedules and is therefore prone to robustness quality and stability degradation when subjected to uncertainty. This main solution method is therefore modified as described in the next section.

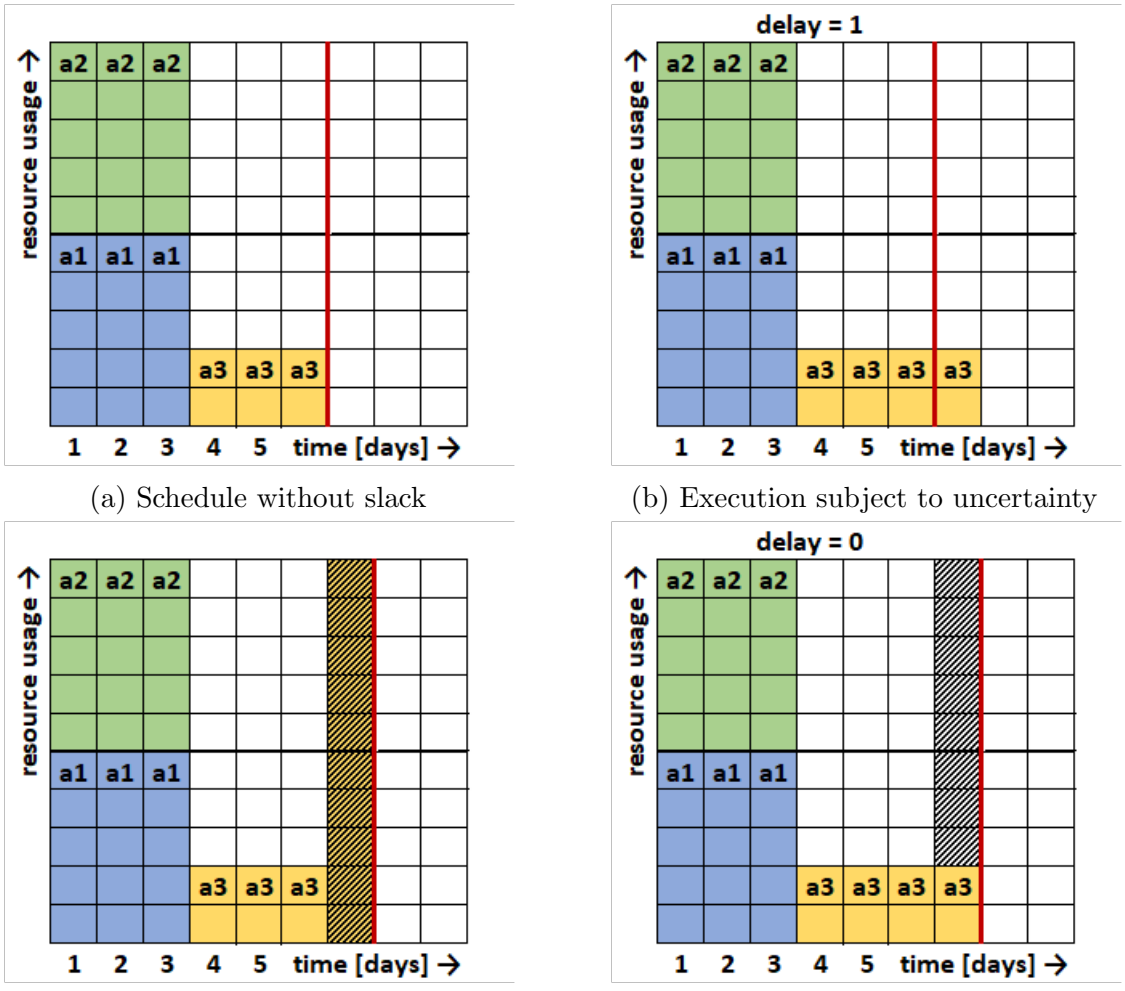
5.3 Improving robustness

Projects are obligated to ensure themselves that their project has a sufficient level of robustness due to the decentralized setting. First, an explanation is given of why it is

difficult to embed a scheduling method in a decentralized system (subsection 5.3.1). Next, the four methods identified in ***RQ4*** are considered to improve robustness (subsection 5.3.2). Unfortunately, none seem to suit the current situation. Therefore, a new algorithm based on the idea of the STC is devised to efficiently improve robustness. The main algorithm is explained (subsection 5.3.3), after which some improvements are presented (Figure 5.4).

5.3.1 Problem

Methods that improve robustness for RCMPSP-instances generally make use of private information and are therefore not suited. When methods are used to improve robustness for RCPSP-instances, new problem dynamics arise, as explained next using an example (Figure 5.2). Consider a single-project environment with project **A** having three tasks in which task a1 and a2 precede task a3. The due date is set at the finishing time of the last scheduled epoch.



(a) Schedule without slack

(b) Execution subject to uncertainty

(c) Schedule with slack

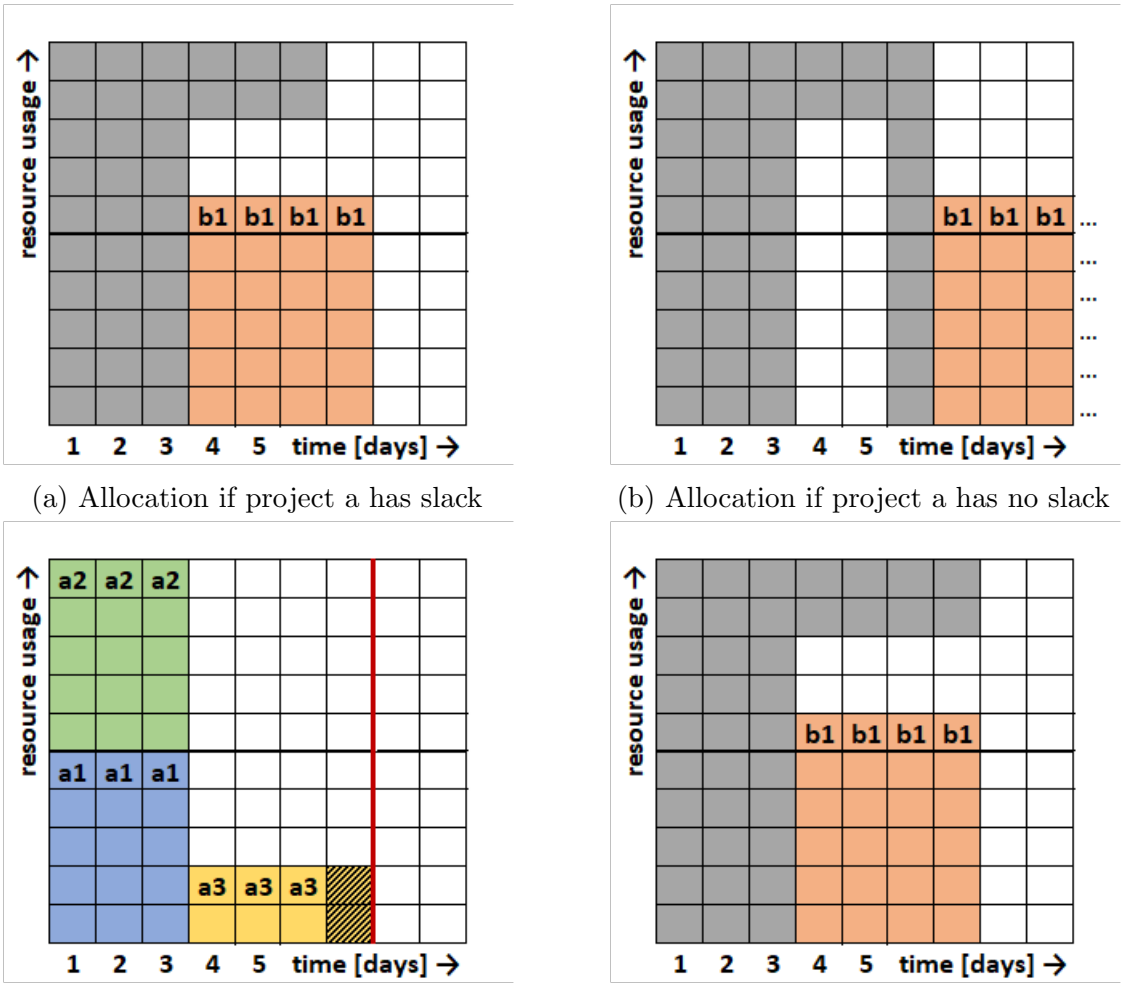
(d) Execution after uncertainty

Figure 5.2: Effect of adding slack in single project environment

No slack is added in the first instance (a). Suppose task a3 is disrupted during execution, in this case the project delay is 1 (b). It is best to add slack for task a3, as it can absorb the slack of itself and its two anterior tasks (c). Slack is indicated using the pattern

overlay. That way, it may look as if those resources are 'reserved', but that is not the case in single-project environment. If a task receives slack, all tasks not yet started (such as the dummy end task) can simply be shifted to the right. It does not matter how many unused resources are required in making a robust schedule since all resources are dedicated to this project. Since slack is added, there is no delay (d). Moreover, regardless of which task is delayed (within the uncertainty that is expected), the project is within the due date. In the illustration, slack is indicated and 'reserved'

This changes however when considering multi-project environments. Unused or unoccupied resources are now valuable as they are available to other projects to schedule their tasks in. Now, a need to reserve resources arises. Only right shifting tasks is not sufficient, since that empty space may be utilized by other projects. Therefore, resource have to reserved. Suppose that project **A**'s resources are allocated in the bidding process. These resources are now subtracted from the availability such that a new project **B**, having task b1, can be scheduled (Figure 5.3).



(a) Allocation if project a has slack

(b) Allocation if project a has no slack

(c) Allocation if project a has slack

(d) Allocation if project a has no slack

Figure 5.3: Effect of adding slack in multi-project environment

In the instance without slack addition, task b1 is scheduled in the first feasible timeslot (a). In a multi-project environment, reserving all slack is inefficient since it blocks many time slots, therefore shifting task b1 (b). Looking at the project instance, only two resource units are required for task a3 to absorb its own uncertainty, hence this is a minimal

amount of resources to reserve (c). Using this policy, project **B** can be scheduled efficiently, whereas project **A** still has sufficient slack to absorb the uncertainty of task a3.

The introduced dynamic is in determining the amount of resources to reserve when adding slack. A maximum amount is all resources available at a time slot. A minimum amount is the number of resources that the task at hand requires. The more resources are reserved for slack, the better the ability to absorb uncertainty, but it blocks at the same time resources that could be used by other projects.

This issue does not necessarily arise in other robust project scheduling methods in multi-project environments. First of all, assuming symmetrically distributed information (centralized), the contents of allocated resources is known in later decision making. Therefore, robust scheduling methods can use the information to account for the impact of scheduling tasks at certain timeslots. An example is buffers between tasks whenever a resource switches to work on tasks of another project. Another approach seen in current methods is to prohibit multi-tasking at a task or project level, as is advised in most circumstances in multi-project CCM (W. Herroelen & Leus, 2007). Multi-tasking at a task level is the execution of two tasks simultaneously within one resource pool. That means that the tasks in a project are executed in series and can only start after the previous finished. This leads to inferior solutions.

Hence, a scheduling method is sought that either deals with RCMPSP-instances without violating private information or that deals with RCPSP-instances without reserving all units of slack.

5.3.2 Consideration of candidate methods

The four methods identified in ***RQ4*** are considered to improve robustness (see also Table 4.2).

The CCM multi-project variant does not seem suitable. First of all, it focuses on the most constrained resource, while multiple resources may be highly constrained altogether. Tian et al. (2019) underlines the importance of considering resource interactions between projects, ignoring it results in reduced effectiveness in solving RCMPSPs. Moreover, multi-project CCM aims to allocate drum buffers to protect the most constrained resource, but in order to do that, private information is required of other projects. Although workarounds could be devised based on assumptions on private information, it would be an uncertain approach.

The CCM single-project variant does not seem suitable. Deblaere et al. (2007) point out that buffers and their sizes may be too abundant, unnecessarily lengthening the project, whereas in other occasions, the buffers may not be able to protect against the propagation of disruptions. Moreover, W. Herroelen and Leus (2007) presents two whole sections on the applicability of CCM in particular situations, among others indicating that CCM may not be suited in situation having high uncertainty as in the case at JIVC. Next to that, the method reserves all units of slack.

The MABO and PSO with chaining as that proposed by Policella et al. (2008), do not seem suitable. Both resource flow network methods are more focused towards optimizing resource allocation and retaining stability as opposed to scheduling and quality robustness. Although positive effects are reported regarding quality robustness, it makes them less

suitable compared to other methods. Moreover, the chaining heuristic used by Policella et al. (2008) relies on the random generation of solutions within its procedure. This violates the requirement of JIVC to have a deterministic solution method. Next to that, the methods reserve all units of slack. A workaround could be devised by attaining those solutions via a predetermined policy, although it will probably result in less optimal solutions.

The STC approach seems most suitable. Experiments in publications on the STC report good robustness improvements for a rather simple heuristic, yet van de Vonder (2006) reports on disadvantages for larger problem instances. Computational effort for larger projects increases, taking too long to execute the heuristic regularly as for example is required in bidding processes. Next to that, the method reserves all units of slack.

The conclusion is that none of the current methods is suitable for direct implementation. The STC measure seems most suitable but is computational inefficient for larger instances. In its algorithm, it calculates the longest path through a graph from the dummy start task to any other task, in each of its iterations. For a project of 120 tasks, this requires about two seconds to calculate a solution according to their findings. This would drastically increase computational time in bidding processes of decentralized methods that call that function to improve robustness. Therefore, a new algorithm is devised based on the approach of the STC measure that does not block all slack and is less computational demanding.

5.3.3 Main algorithm

The STC measure inserts buffers on the most suitable places based on start time criticallity. The CSS measure is an estimator of robustness and is developed after the existence of the STC. The CSS could be used as a replacement for the start time criticallity surrogate measure to 1) decide on the amount of slack to reserve and 2) to reduce the computational effort required. In this section, an algorithm is devised that employs the CSS to improve the robustness of a schedule.

Both algorithms start with a generated schedule of a single project. First, the CSS is determined which is an estimator for its robustness. Robustness may be improved by adding a certain amount of slack to the end of a certain non-dummy task of the project in iterations. The CSS can be recalculated after addition of an amount of slack to a task to determine the improvement in robustness. A number of problems arise.

Finding the best improvement

The first problem is finding the (task, slack amount)-combination that increases robustness most. The most straight forward approach is to iterate over all non-dummy tasks, add slack to that task, reschedule the complete instance to account for the increased duration of the task, and measure the CSS. Next, the improvement can be calculated by taking the difference between the old and new schedule and dividing it by the added slack amount. This results in the estimated robustness improvement per unit of slack added, which is referred to as the 'gain'. The best (task, slack amount)-combination is the one having the highest improvement per unit of slack.

Deciding on the range of slack

The second problem is deciding on the range of amounts of slack to add to each task in each iteration. after which the robustness should be determined.

Only adding just a single unit at a time is a non-optimal approach. For example, adding a single unit of slack may be insufficient to increase robustness since it is not enough to absorb the expected uncertainty. Adding two units of slack to the same task may be sufficient to absorb uncertainty and greatly increase the robustness. If only one unit of slack is added at a time, some preferable robust solutions are missed because the first decision will never be made. Therefore, a range of slack amounts from 1 to a certain amount should be tested.

Observing how the CSS is calculated, it can be concluded that there is a maximum amount of slack that can be added after which no improvement is possible. This is the moment that every task has an amount of slack that it is sufficient to absorb its own expected slack and that of all anterior tasks (see section 4.1.1). That amount could be used as an upper bound. Unfortunately, this value is not fixed but schedule dependent, because the tasks that are anterior are schedule dependent. Hence, it cannot be determined. Yet, the longest occurring task duration can be selected, and its sufficient slack is a valid upper bound.

Deciding on a gain based termination criteria

The third problem is deciding on a termination criteria to prevent an increase in makespan when the additional robustness is not worth it anymore. A fixed amount of addition in robustness per unit of slack could suffice. Unfortunately, values of the CSS are instance dependent. Suppose a certain robustness value of schedule that is made robust, for example 300. This may be a high robustness value in a small instance giving great robustness, while being not robust at all in a large instance. Therefore, it is difficult to decide which improvement per unit of slack is no longer worth the cost in makespan. By means of experimentation, it was found that gain greater or equal to four is worth it and hence chosen as threshold. This is however a trade-off problem depending on user preference, therefore the gain is made an adjustable parameter.

Deciding on an absolute termination criteria

The fourth problem is deciding on a termination criteria to prevent a too high absolute loss of makespan. For example, a schedule may still be able to be improved based on CSS gain, but require too much makespan. Therefore, an absolute upper bound is set, which is equal to 10% of the project makespan (rounded) of the input schedule. The decision for 10% is based on the expected uncertainty, which is a duration increase of 10% for 10% of the tasks. Therefore, a maximum increase of 10% seems reasonable. This termination criteria is also depending on the user preference and made as a parameter.

Deciding on the amount of resource to reserve

The fifth problem is the amount of resources to reserve per resource level when adding slack. For now, that resource level is set equal to the resource requirement of the task, such that there are always sufficient resources reserved such that the task itself can make

use of the slack, such that other projects may use the remaining resources to schedule their tasks.

Algorithm

The algorithm, encoded in Python, is presented Listing 5.1. The input of the algorithm consists of 1) an RCPSP problem instance object, 2) a feasible schedule of that instance, 3) a gain threshold, and 4) an absolute threshold. The output is a 1) problem instance object, and 2) a schedule. The problem instance is an object that has a set of tasks. Each task is an object with functions to add and remove slack. The algorithm also contains functions to generate a schedule and calculate the CSS. The output also contains the problem instance, because it is slightly modified, since it registers the slack that was added to the task object.

```
remaining_slack = absolute_threshold
css = calculate_css(problem_instance, schedule)
while remaining_slack > 0:

    # Generate all candidate options
    candidate_options = []
    for task in problem_instance.non_dummy_tasks:

        # Determine the range to add slack on
        maximum_slack_addition = min([remaining_slack, css_longest_task])
        slack_range = range(1, maximum_slack_addition + 1)

        for additional_slack in slack_range:

            # Modify instance, generate schedule, calculate new css
            task.add_slack(additional_slack)
            modified_schedule = generate_schedule(problem_instance)
            new_css = calculate_css(problem_instance, modified_schedule)
            task.subtract_slack(additional_slack)

            # Check if it is a valid candidate
            delta_css = new_css - css
            if delta_css > gain_threshold:
                gain = delta_css/additional_slack
                candidate = (gain, additional_slack, task)
                candidate_options.append(candidate)

    # If there is at least 1 candidate, select the best one, else terminate
    if candidate_options:
        chosen_candidate = max(candidate_options, key=gain)
        _, additional_slack, task = chosen_candidate
        task.add_slack(additional_slack)
        schedule = generate_schedule(problem_instance)
        css = calculate_css(problem_instance, schedule)
        remaining_slack -= additional_slack
    else:
        remaining_slack = 0
return schedule
```

Listing 5.1: Python pseudo-code of robustness improvement algorithm

Performance

Experiments using this algorithm show that the performance is similar to that reported for the STC, which is regarded insufficient for larger .

5.3.4 Algorithm improvements

The first algorithm requires considerable computational effort. In each iteration, all tasks are evaluated, and for each task, a schedule is generated for each slack amount on the range $[0, min(remaining\_slack, css\_longest\_task)]$. This becomes computational problematic when the number of tasks and range increase. The computational efficiency can be improved at the cost of possibly attaining a less optimal solution.

Instead of generating a schedule for each possible task and slack amount, new candidates are selected without generating a new schedule. This reduces the computational effort, but it is an approximation since it contains a possible error. Adding slack to a schedule increases duration and might lead to a different schedule due to being infeasible or a different selection by a priority rule. In that case, tasks may be scheduled differently from the original scheduled epoch of the disrupted task onwards. The CSS measure looks backwards to anterior tasks. Hence, the error when not generating a new schedule is contained at the scheduling epoch and is only large if the new schedule is different. At the end of each iteration, the schedule of the best candidate is generated, therefore always resulting in feasible solution. Based on experiments, the approximation yields good solutions while being significantly faster. The line of code on line 17 is removed completely.

Furthermore, from experiments it is found that the added slack rarely exceeds two units added in a single iteration. Therefore, evaluating a high range of slack amounts is unnecessary. An additional upper bound of 4 is included. The evaluation on line 10 is modified to the range $[0, min(remaining\_slack, css\_longest\_task, 4)]$ by including 4 in the min() function.

5.4 Modifications

The replicated method is modified and a high level overview of the modified agent interaction is presented (Figure 5.4).

5.4.1 Improving single schedule robustness

The algorithm to improve robustness as presented in the previous section is included in the bidding step and bid modification step (red bar).

5.4.2 Include slack in bids

Slack usage should be communicated to the auctioneer. Task information cannot be communicated, therefore, slack cannot be communicated as being part of a task. Furthermore, slack cannot be added to the resource usage, even though it is a reserved slot, because that would influence other performance measures. Therefore, an additional resource usage

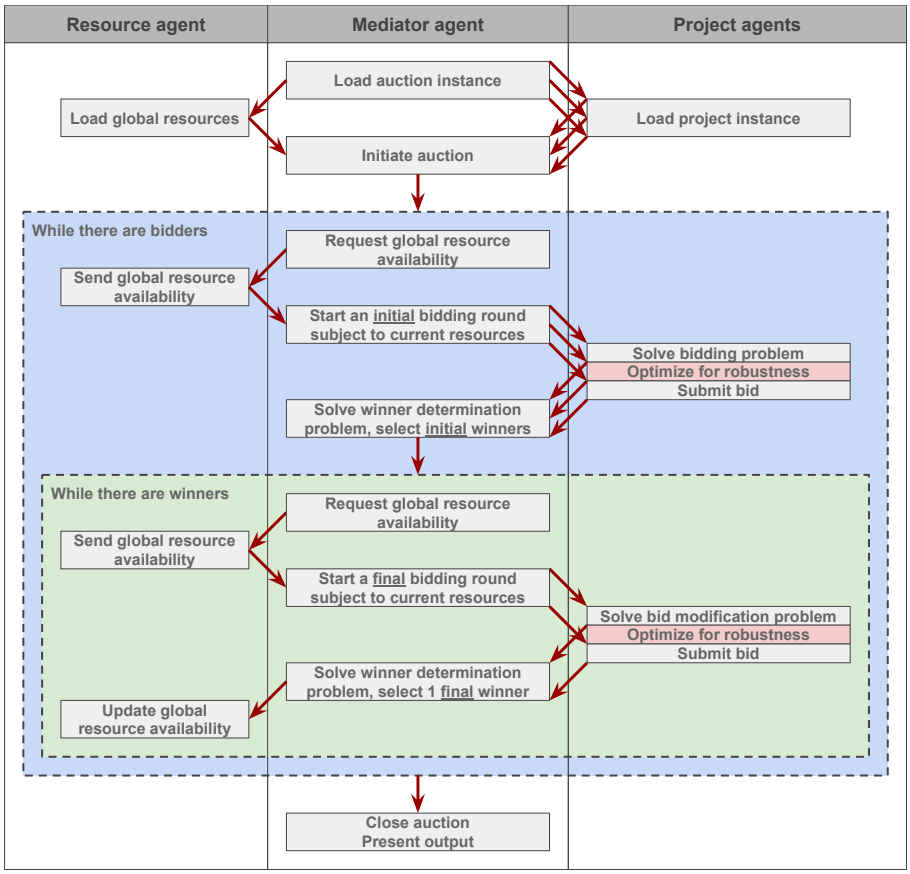


Figure 5.4: Abstracted agent interaction of modified method

profile for the slack is added to each bid similar to how resources are shared. This preserves private information.

5.4.3 Continuation of auction

The role of the auctioneer does not change. Therefore, just as regularly, an auctioneer considers initial bids, calculates the demand ratio, chooses initial winners, asks project agents for a final bid given a demand ratio, determines actual winners and allocates resources to the actual winners.

5.4.4 Output

The output is similar to the output of the replication. The difference is in the added slack in schedules, as can be seen in the example Appendix K.

5.4.5 Settings

Previously, the input variables were presented. Next to these input variables, there are settings and additional parameters that can be regarded as secondary input variables.

- The scheduling mode of the schedule generator can be changed and set for each project individually. For example, a serial or parallel generation scheme having a certain priority rule can be set.

- The sufficient slack (or expected slack) can be set on a global level. With a slight modification, it can be made project specific.
- The robustness improvement thresholds can be adjusted.
- All the output generation can be turned off or on, including image plotting and .txt document generation, except for the logging file.
- A settings is concluded to set the unit on the horizontal time axis (for example 'days' or 'weeks') and on the vertical axis (for example 'hours' or 'FTE').
- The resource utilization plot shows an average using its closest neighbours. The number of neighbours used in averaging can be set.
- The plots can be adjusted, for example font size, image scaling, output type (.pdf, .png, .jpg, etc.), colour map/schemes, etc.

5.5 Capabilities

This section presents the capabilities and possible use cases of the RDPSM.

- **Project scheduling**: First and foremost, the RDPSM produces schedules for RCMPSP problem instances such that project stakeholders know when to start tasks.
- **Resource management**: The RDPSM output contains both global and local resource utilization charts such that project stakeholders can immediately identify under- and over-utilization given the current input.
- **Project crashing and hiring**: The RDPSM generates critical schedules per project, giving project stakeholders insight in the shortest possible makespan (given infinite resources) and the maximum required resources to achieve that makespan. This may assist project stakeholders in decision making regarding project crashing and/or hiring of external employees.
- **Due date estimation and forecasting**: The generated robust schedules provide (internal) due dates for project stakeholders. These (internal) due dates can be turned into (external) due dates to communicate towards customers.
- **Resource allocation**: The RDPSM allocates resources by accounting for costs and revenues, and additionally the temporal (or scheduling) aspect. This is an aspect that is currently very difficult to account for by project stakeholders.
- **Portfolio selection**: A project stakeholders composing a project portfolio can see how certain compositions behave when taking into account the temporal and resource aspect. This is an aspect that is currently very difficult to achieve.
- **Scenario analysis**: For all of the preceding points, project stakeholders may run different scenarios by changing the input variables and analysing the effects. This enables project stakeholders to answer 'what if...?' questions which greatly assists in decision making.
- **Advice and comparison**: Planners at JIVC currently employ their own scheduling strategy. The RDPSM does not have to necessarily replace their scheduling strategy, it could also be used to present schedules that DMUs at JIVC can use as comparison to their own methods and improve them.

5.6 Identified improvements

Multiple improvement options are identified throughout the thesis. These improvements are not executed due to a multitude of reasons, such as having a low priority or having been identified too late during the execution of the thesis to be improved in time.

Computational efficiency

The computational efficiency can be significantly increased. As indicated before, most efficiency can be gained by implementing parallel execution or multi-threading of the bidding process, the bid modification process and the improving robustness process. Furthermore, Python provides libraries that utilize the GPU such as the CUDA and Numba libraries and some functions in the TensorFlow library. Additionally, memory usage can be reduced by storing resource usages and multisets as sparse matrices (since most values over the time horizon are zero). If efficiency is a real issue, other programming languages are preferred that are known to be more efficient, such as C++.

Characteristic based execution

As indicated before, performance depends on the project instance characteristics. These characteristics can be calculated before initiating the scheduling process. Depending on the measured characteristics, the main method could choose to run the algorithm using different scheduling methods and parameter configurations. For example, another priority rule may be chosen for schedule generation, or the

Post-auction improvement

After the final bid was accepted, an additional and easy-to-implement step could be included. Each project manager could optimize its own schedule for the highest robustness (i.e. stability) by moving task with free slack to other feasible starting times. Project agents previously moved tasks depending on the demand ratio and as such avoid slots in high-demand. After the final bid, each project agent was allocated its resources, so the starting times can be optimized for robustness.

5.7 Conclusion RQ6

To conclude this chapter, a summary is given of the answer to the research question.

RQ6: An RDPSM is designed by selecting the DRCMPSP MAS of Song et al. (2016) as a core framework and making it robust. The robust methods identified in ***RQ4*** are not suitable for implementation in the MAS. To improve robustness, a buffer insertion algorithm is designed inspired by the STC approach in combination with the CSS surrogate robustness measure by Khemakhem and Chtourou (2013). The core framework is modified to include the algorithm.

Chapter 6

Treatment validation

In the previous chapter, an RDPSM is designed. In this chapter, ***RQ7*** is answered by validating the method using as input the MPSPLib problem instances. First, the effects of the method are evaluated by running the replication and the designed RDPSM (section 6.1). Next, the performance of the RDPSM is evaluated by means of a Monte-Carlo simulation. First, the set-up of the experiment is presented (section 6.2) after which the results are shown (section 6.3).

The experiments are performed on a Lenovo Thinkpad P1 laptop with an Intel® Core™ i7-8750H CPU Processor at 2.20 GHz, having 6 cores and 12 logical processors on a system with 16 GB installed RAM. As explained, the code is not optimized for efficiency. Also, multiple demanding background programs ran during the experiments. Therefore results on computational times should be regarded as an upper bound.

6.1 Effects

This section presents the impact of the robustness improvement algorithm. The instances of the MPSPLib dataset are processed by the replication and the RDPSM to compare the effects. The full results are presented in Appendix M. The MPSPLib dataset consists of multiple similar instances that can be grouped into sets based on the number of projects and task per project. This is a regular practice in literature that use the MPSPLib.

6.1.1 Comparison of publication, replication and R&DPSM

First, a comparison is made between 1) the publication by Song et al. (2016), 2) the replication of that publication in this thesis and 3) the designed RDPSM (Table 6.1). The comparison is made based on the Average Project Delay (APD), since that is what Song et al. (2016) report on.

The first conclusion is that findings and performance of Song et al. (2016) could not be replicated, which is an unexpected result. The recreated method without robustness modification consistently performs worse compared to the reported results of Song et al. (2016). On average, the solutions produced by the replication have an average project delay that is 10,59 days longer, a 13,05% degradation of the makespan. Possible causes are as follows. The exact parameter configuration may deviate since it is not specified in

Table 6.1: Comparison of average project delay for each of the problem instance sets

	Average project delay		
Instance set	**Publication**	**Replication**	**RDPSM**
30_2	13,60	19,30	23,20
30_5	19,79	23,37	27,60
30_10	55,78	59,82	67,72
30_20	116,10	124,31	134,54
90_2	6,00	13,30	18,90
90_2_AC	108,15	125,75	133,15
90_5	10,72	13,96	18,48
90_5_AC	249,42	278,46	293,82
90_10	39,02	44,42	50,96
90_10_AC	175,23	189,27	203,86
90_20	20,95	24,15	29,65
90_20_AC	94,05	96,79	104,82
120_2	50,60	67,00	70,00
120_2_AC	37,75	53,60	64,45
120_5	45,92	60,28	67,40
120_5_AC	181,30	205,58	224,10
120_10	107,14	116,40	126,08
120_10_AC	103,74	112,37	121,87
120_20	24,27	35,62	40,86
120_20_AC	163,36	170,90	184,88
Average	**81,14**	**91,73**	**100,32**

Song et al. (2016). Some specifics in the algorithm are not specified, for example, how tie-breaking is resolved in selections. Additionally, it cannot be ruled out that errors have been made in coding.

The second conclusion is that the schedule improved for robustness has an increased makespan, which is as expected since it is a trade-off problem. On average, the makespan of the schedules produced by the RDPSM are 8,58 days longer compared to the replicated method, which is a 9,36% increase.

6.1.2 Comparison of replication and R&DPSM

Song et al. (2016) do not report on other measures in the same way they do as the APD in the previous section, therefore the next comparison only regards the replication and the RDPSM. (Table 6.2).

The following is concluded based on the results. The average makespan (AMS) increases by about 5%, and the total makespan (TMS) increases by about 5%. These are expected results, since projects require more time to be completed due to the added slack that should absorb robustness. This means that even though each project is delayed by on average 5%, the duration of the whole portfolio increases only by 5%. This would be higher if all projects were executed one after another, therefore it can be concluded that there is a considerable parallel execution (or overlap in projects).

Table 6.2: Percent change in output after robustness modification

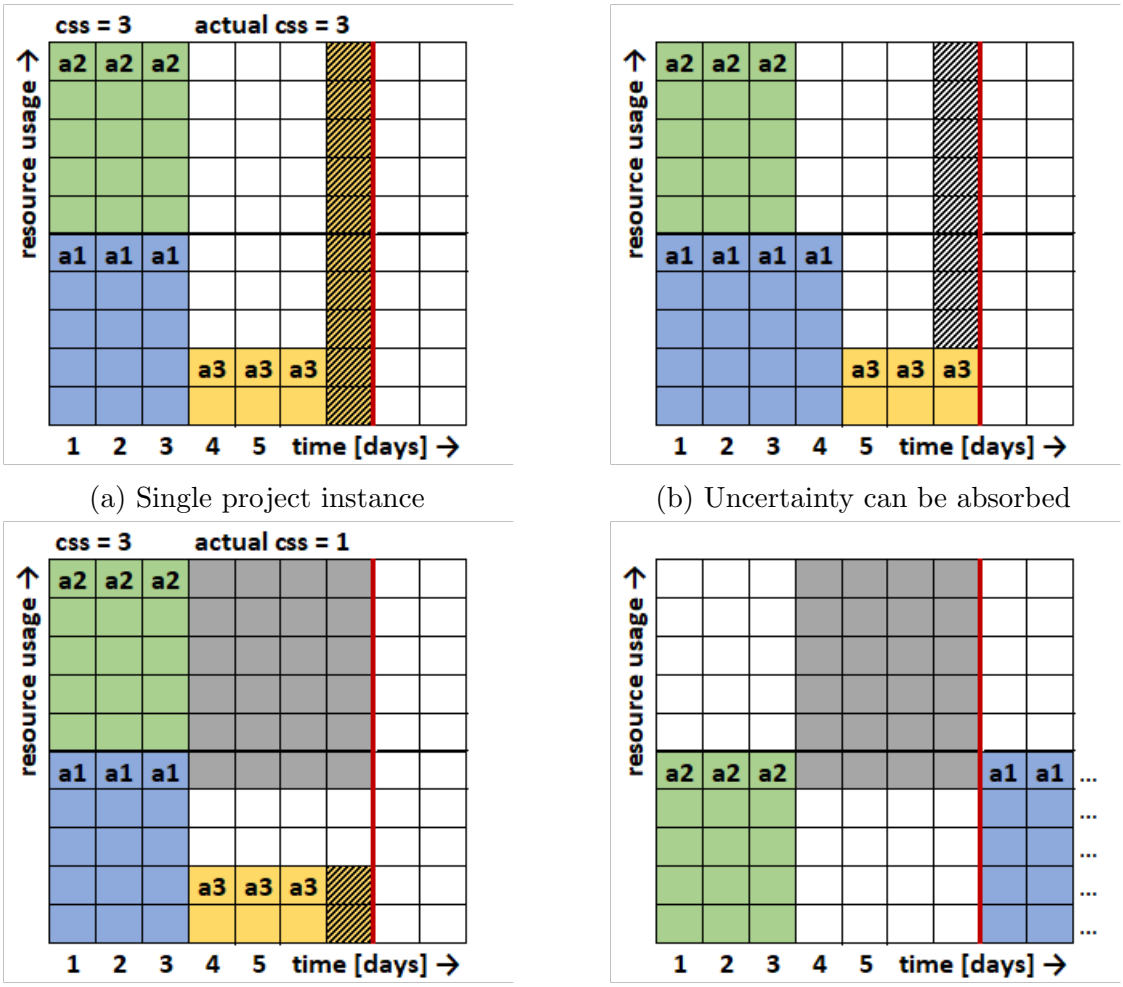
	Relative difference [%]			
Instance set	**AMS**	**TMS**	**APD**	**DPD**
30_2	5,85	5,54	20,21	3,56
30_5	5,76	8,91	18,12	7,40
30_10	7,34	5,56	13,21	9,28
30_20	5,76	6,42	8,23	6,08
90_2	5,26	5,50	42,11	15,18
90_2_AC	3,64	4,93	5,88	9,64
90_5	4,46	3,63	32,38	3,11
90_5_AC	4,32	3,73	5,52	4,28
90_10	4,93	5,10	14,72	2,12
90_10_AC	5,44	5,58	7,71	4,60
90_20	4,78	0,63	22,77	-1,91
90_20_AC	4,56	5,21	8,29	4,70
120_2	1,78	-0,78	4,48	-13,68
120_2_AC	5,77	6,07	14,65	8,24
120_5	4,71	5,17	11,81	5,46
120_5_AC	6,28	6,38	9,01	6,61
120_10	4,64	4,06	8,32	3,11
120_10_AC	4,62	6,08	8,45	5,70
120_20	4,10	5,50	14,71	2,51
120_20_AC	5,23	4,92	8,18	5,41
Average	**4,96**	**4,91**	**13,94**	**4,57**

The average project delay (APD) increases by about 14%. This indicates that the previously 5% increase in average makespan represents a project delay that is three times as high. This may be misleading because the due dates in this project are set to the critical path duration, a date that is infeasible. This sheds a more pessimistic view. More meaningful is the standard deviation of the project delay (DPD). This measure increases on average by 5%, meaning that within an instance, there are some projects that have a dis-proportionally larger delay compared to others.

6.1.3 Observed limitation of robustness improvement

As indicated in subsection 5.3.1, deciding on the number of resources to reserve when adding slack is a problem dynamic arising in adding slack in multi-project environments. Reserving all resources at an epoch for slack is non-optimal and blocks other projects from using valuable resources. Therefore, the current algorithm is set to a 'minimum resource selection' policy such that it blocks a minimum number of resources required by the task that receives slack. Unfortunately, the CSS measure gives an incorrect robustness measurement using that minimum resource selection policy. To show this deficiency, an example is provided (Figure 6.1) which is a continuation of that in subsection 5.3.1.

Consider the single-project variant (a). One unit of slack is added to task a3. Assuming that this unit is sufficient, the CSS measure is three, because task a3 can absorb uncertainty of task a1, a2 and a3. Hence, the actual CSS is also three. For example, consider



(a) Single project instance

(b) Uncertainty can be absorbed

(c) Multi project instance

(d) Uncertainty cannot be absorbed

Figure 6.1: Effect of adding slack in multi-project environment

that task a1 is delayed by one unit (b). As indicated before, all resources are blocked in the single-project variant, which is inefficient in a multi-project setting. Therefore, the robustness improvement algorithm only reserves as many units as the task requires resources. Consider the multi-project variant where both project **A** and **B** are scheduled, the CSS measure again indicates a value of three (c). This is incorrect, as the limited resources due to project **B** block the ability of task a3 to absorb its uncertainty. For example, if project a1 is delayed by 1 unit, the task must be rescheduled to t=8, and task a3 as a successor of a1 is rescheduled to t=11 (d). Therefore, it a3 can only absorb the uncertainty of one task instead of three, giving an incorrect measure.

The first insight is that the measure could be fixed by considering the amount of resources to add as a variable. In that case, the CSS is adjusted such that a robustness score is only appointed if the task also has sufficient resources. The second insight is that the solution space significantly increases because of this new degree of freedom in the measure. Not only should each amount of slack be considered for each task, also the amount of resources to add should be considered. This is undesirable.

The third insight is a proposed solution. Set the amount of resources to the maximum resource requirement from all the anterior task and itself for which it has sufficient slack.

This way, the minimal amount of slack is chosen such that the CSS measure is always correct. Since this insight was obtained after completion of the experiments, the difference in its performance is unknown.

Deciding on the amount of resources to reserve is still a trade-off problem. Selecting all resources is the most conservative measure but may unnecessarily reserve resources. Selecting only the required resources of that task may be too optimistic and degrade robustness. Still, robustness is now improved based on the surrogate measure, which gives incorrect values under the current policy. Yet, remind that the example above is an explicit one. It is unknown how often it occurs.

6.2 Experiment set-up

The performance of individual schedules is estimated by the surrogate CSS measure of Khemakhem and Chtourou (2013). This does not estimate the performance of the multi-project instance, therefore it cannot be used to estimate or evaluate the performance of the method. An exact evaluation is intractable, therefore simulation is the only remaining approach for evaluating robustness. The performance of the RDPSM is determined by subjecting the solutions as generated by the RDPSM to disruption scenarios. First, the scenario generator is presented (subsection 6.2.1). Next, the simulation is explained (subsection 6.2.2). This simulation requires a rescheduling method. The impact of this rescheduling method is presented subsequently (subsection 6.2.3).

6.2.1 Scenario generator

The scenarios that occur are simulated by a generator (Figure 6.2).

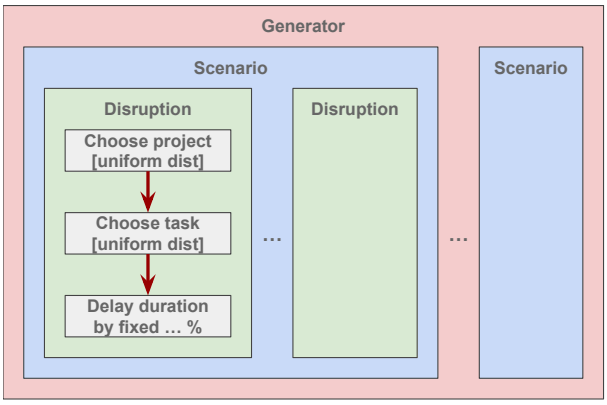


Figure 6.2: Set-up of treatment validation experiment using a Monte-Carlo simulation

The generator takes as input a problem instance, for which it generates a user defined number of scenarios that subject the input to uncertainty. Each scenario consists of a user defined number of disruptions (set to 10% of the total number of tasks in the problem instance). A random project is chosen from the project agents in the instance using a uniform distribution. Subsequently, a random task is chosen from the tasks of that project agent (excluding the dummy start and finish tasks) using a uniform distribution. Hence, a disruption consists of a unique (project, task)-combination, and the duration of that task is increased by user defined amount (set to 10%). A newly generated

combination is compared to previously generated combinations, such that doubles are omitted to guarantee that all combinations in a disruption are unique. A newly generated scenario is sorted and compared to previously generated scenarios, such that doubles are omitted.

The generator therefore generates (or chooses) samples from the population that consists of all possible combinations of disruptions. When a solution is subjected to uncertainty, the results (performance measures) only apply to that sample and not to the whole population. Therefore, results are presented as confidence intervals. For this simulation, 95% confidence intervals are used. Additional information on the population size and chosen sample size is presented in Appendix L.

6.2.2 Monte-Carlo simulation

A Monte-Carlo simulation (or stochastic simulation) is employed to subject the schedules to uncertainty using the scenario generator. The set-up of the treatment validation is visualized (Figure 6.3).

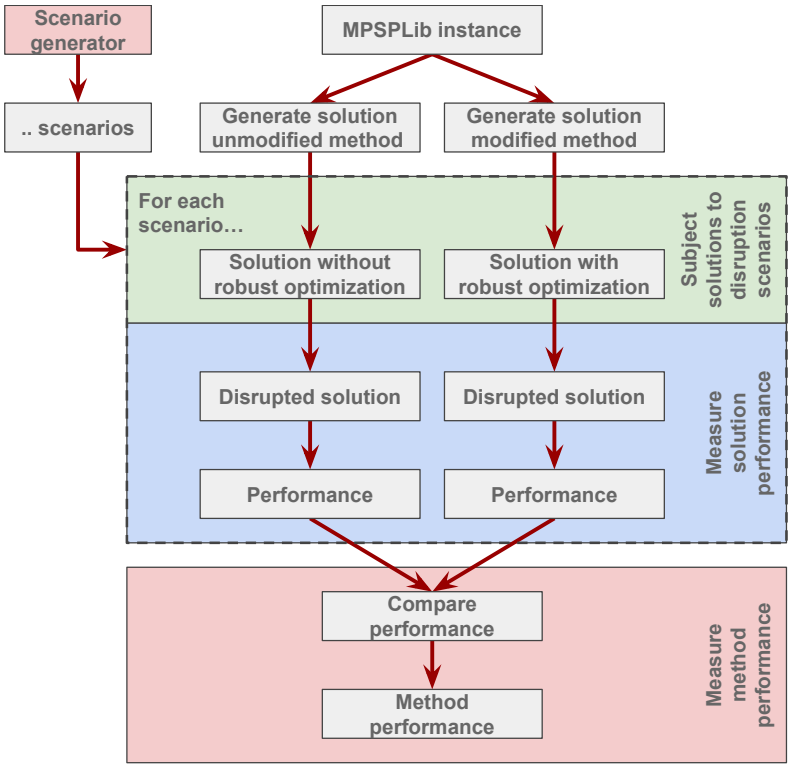


Figure 6.3: Set-up of treatment validation experiment using a Monte-Carlo simulation

For each problem instance of the MPSPLib, a pair of two solutions is generated using the RDPSM. The first solution is generated without using the modification of the previous chapter, hence it is not optimized for robustness, whereas the second solution is optimized for robustness.

For each schedule, the awareness epochs are determined for each disruption in the scenario. As indicated before, each disruption has an awareness epoch which is set equal to the scheduled start epoch of that task. Hence, the awareness epochs can only be determined when having a solution.

Virtual execution

Next, that solution is 'virtually' executed. Tasks are executed according to their scheduled start times. Tasks may be able to start earlier due to build-in robustness but this is not allowed, following a 'railway execution' regime. The time starts at 0 and is forwarded until an agent is aware of a disruption, which is the minimum awareness epoch out of the set of disruptions. All tasks initiated before that epoch are executed or in execution and cannot be changed. Multiple disruptions could occur at the same epoch. In that case, the following order is chosen: The project agent who won earlier in the bidding process resolves its disruption first. Still, that agent may have multiple disrupted tasks starting at the same epoch. The task having the smallest index number is resolved first. The disruption has to be resolved, and a three step plan is implemented to do that.

Resolving problems

First, a check is performed to see if the disrupted task has sufficient slack to absorb the uncertainty. If that is not the case, a check is performed to see if there are sufficient available resources to absorb the disruption. If that is not the case, all remaining tasks are rescheduled for this project. All the slack of the remaining tasks are removed, and the tasks are rescheduled given all available free resources using the MINLFT SGS which is also implemented in the method.

6.2.3 Impact of rescheduling on validation

The simulation requires a rescheduling method. As indicated by E. Demeulemeester and Herroelen (2010), the measured performance depends on the rescheduling method and the quality of repaired schedules. Rescheduling methods may change the start times of tasks, leading to instability and nervousness. Therefore, the performance of the RDPSM using simulation depends on the rescheduling method that is employed. The selection of a suitable rescheduling method is a well-known problem in verifying performance.

Rescheduling methods work completely different as opposed to pro-active scheduling methods, see also subsection 2.1.3. The need to select the most suitable rescheduling method is unexpected, and the selection of the most suitable rescheduling method to evaluate the RDPSM is out of scope of this thesis. Therefore, as mentioned in the previous section, a relative simple MINLFT SGS is implemented as a rescheduling method. The author is well aware that this negatively impacts the validation of the RDPSM.

6.3 Experiment results

This section evaluates the performance of the RDPSM. The schedules generated by the replicated method and the RDPSM are both subjected to the same disruption scenarios. Two performance measures are evaluated, being 1) the percentage of cases that the instance finishes within the scheduled due date and 2) the makespan in which it was completed. The full results are presented in their 95% confidence intervals in Appendix N. These full results are aggregated per instance set similar to the tables in section 6.1. First, in-time delivery results are inspected (Table 6.3).

The table is explained using the first instance set having 2 projects with 30 tasks per

Table 6.3: In-time delivery performance between non-robust and robust method

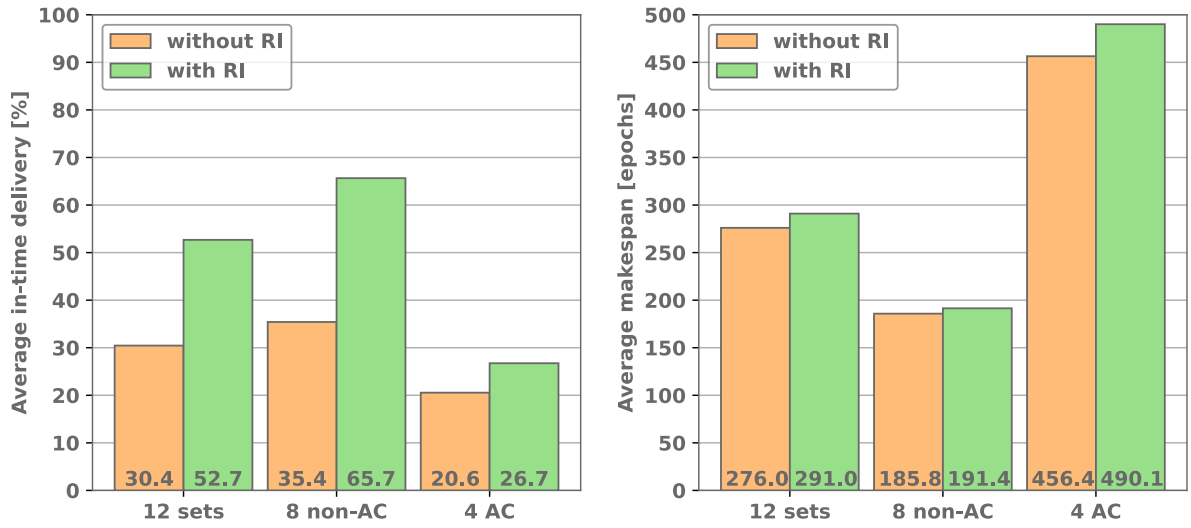
	In-time delivery [%]		Difference	
Instance set	**Replication**	**RDPSM**	**Absolute**	**Relative**
30_2	37,68	84,20	46,52	123,46
30_5	53,67	85,65	31,98	59,60
30_10	14,18	28,50	14,32	100,99
30_20	2,66	9,96	7,30	274,44
90_2	60,74	97,74	37,00	60,92
90_2_AC	17,77	19,80	2,03	11,40
90_5	21,82	63,72	41,90	192,03
90_5_AC	4,72	5,52	0,80	16,95
90_10	40,82	83,32	42,50	104,12
90_10_AC	12,80	17,66	4,86	37,97
90_20	51,68	72,12	20,44	39,55
90_20_AC	46,89	63,97	17,08	36,43
Average all	30,45	52,68	22,23	72,99
Average non-AC	35,41	65,65	30,25	85,42
Average AC	20,55	26,74	6,19	30,14

project (first column). Schedules generated by the replicated method finished within its makespan in on average 37,68% of the disruption scenarios (second column). Schedules generated by the RDPSM finished within its makespan in on average 84,20% of the disruption scenarios (third column). Therefore, the absolute increase in in-time delivery is the difference between the two, being 46,52% (fourth column). The relative increase is therefore 123,46% (fifth column).

The total results are summarized in (a) of Figure 6.4. In it, the replication is indicated as without Robustness Improvement (RI) and the RDPSM as with RI. On average, the replication delivered 30,45% of instances in time, whereas the RDPSM achieved this for 52,68%. Therefore, the schedules generated by the RDPSM are 22,23% more reliable compared to the non-robust schedules, which is an improvement of 72,99%. However, the performance of a scheduling method is often dependant on problem instance characteristics as indicated in previous chapters. The AC (or AgentCopp) instances are more constrained compared to non-AC instances. From the results, it is easily seen that the RDPSM performs better for the less constrained non-AC instances. When splitting the results in a AC and non-AC set, the difference is clear. For non-AC instances, the schedule is on average 30,25% more reliable compared to just 6,19% for non-AC instances. This indicates that the RDPSM is more effective in dealing with less constrained instances in comparison to more constrained instances.

The current problem is a trade-off problem, therefore, makespan is offered for the increase in reliability. Hence, the degradation of the makespan is analysed next (Table 6.4).

The total results are summarized in (b) of Figure 6.4. The average total makespan of all instances using the replication method is 276,03, whereas it is 290,99 for the RDPSM. Therefore, the improvement of in-time delivery comes at a cost of on average 14,96 days, which is a % 5,42 increase of the makespan. The difference between AC and non-AC instances is also prevalent, the makespan is on average increased by only 5,62 days for AC



(a) In-time delivery

(b) Makespan

Figure 6.4: Average results of instance sets, AC-sets and non-AC sets

Table 6.4: Makespan performance between non-robust and robust method

	Makespan [epochs]		Difference	
Instance set	**Replication**	**RDPSM**	**Absolute**	**Relative**
30_2	80,81	82,04	1,23	1,52
30_5	115,75	123,18	7,43	6,42
30_10	217,33	227,87	10,54	4,85
30_20	398,07	422,98	24,91	6,26
90_2	118,15	119,60	1,45	1,23
90_2_AC	279,26	304,34	25,08	8,98
90_5	141,79	142,44	0,65	0,46
90_5_AC	660,77	718,85	58,08	8,79
90_10	223,41	223,38	-0,02	-0,01
90_10_AC	557,80	594,56	36,76	6,59
90_20	191,30	190,03	-1,27	-0,66
90_20_AC	327,93	342,65	14,72	4,49
Average all	276,03	290,99	14,96	5,42
Average non-AC	185,82	191,44	5,62	3,02
Average AC	456,44	490,10	33,66	7,37

instances compared to 33,66 for non-AC instances.

A challenge of scheduling in general is dealing with characteristics differences between project scheduling instances. Various sources found that the performance of RCPSP and RCMPSP solution methods are correlated to the characteristics of their instances, such as Eynde and Vanhoucke (2020). They also report on the effectiveness of a certain scheduling policy, such as that of SGSs priority rules, the use of (double) justification, and the use of robustness measures. Unfortunately, most indicate that performance wildly differs depending on the specific circumstances. For example, Z. Chen et al. (2018) reports that (double) justification is great in deterministic scheduling but has almost no effect in

stochastic scheduling. In another example, Chtourou and Haouari (2008) reports that only specific priority rule and robustness measure combinations lead to effective robust schedules, whereas other even lead to less robust schedules. The constrainedness of global resources is one of the most influential characteristics that effect the performance of methods using RCMPSP instances. The exact same is found in these experiments through the differences in AC and non-AC instances.

6.4 Implementation plan

This section indicates how the RDPSM can be implemented in the processes at JIVC. There are three levels. The first contains an implementation at a practical level in the current situation in which the RDPSM can only be used manually (subsection 6.4.1). The RDPSM can be further integrated into the company such that planning can happen at higher degrees of automation, ranging from a minimal level of automation (subsection 6.4.2) to a complete level (subsection 6.4.3).

6.4.1 Manual use

Currently, employees at JIVC can manually use the RDPSM. They need to manually create problem instances. Although functional, it is rather impractical. The following steps are minimal changes that improve usability and therefore assist in a practical implementation.

Step A, Improve data input: Instances are described in the format as described by MPSPLib. It is not convenient to manually describe an instance in this format. Moving towards a more user friendly format would lower the bar to use the RDPSM. A simple Excel template would be a minimum, low effort, improvement. This should not be an obstacle, as the format of the MPSPLib and describe in Appendix H can be used as example.

Step B, Improve parsing: The current parsers are created to parse MPSPLib instances. The parsers are simple functions, therefore they can easily be swapped for other functions that import more user friendly formats as suggested in the previous step. This should not be an obstacle as long as the programmer has basic Python knowledge.

Step C, Coding routines: Depending on the user and use case (see section 5.5), additional routines could be coded to simplify manual routines that are still required at the moment. For example, automatically configuring settings, running multiple instances, trying different scenarios, etc. Similarly to the previous step, no obstacles are expected given sufficient Python knowledge.

Step D, Guide: A user manual or guide is required for users to work with the RDPSM. Think of proper instance formulation, settings configuration and output interpretation. This may pose an obstacle, as sufficient knowledge is required regarding the functionality of the RDPSM.

Step E, Output: The current output consists of static images and text files. This is informative but not attractive. Simple changes improve the look and feel of the output and increases engagement with users. A Plotly script is already produced in this thesis

that replaces the static MatplotLib imagery and generates interactive schedules accessible in a web browser. Again, this step should not be an obstacle.

Step F, Input data gathering: As of yet, the input data is unsuitable and data has to be manually gathered. Some of the data could be estimated or based on assumptions, but not all. Anyhow, the data is scattered over different employees, and their cooperation is required to attain the required data. This is an obstacle and could be an exhaustive task, since their compliance and availability is required while they are all busy doing their day-to-day job. Although not part of this range of steps, it underlines the need to reduce dependency on employees for access to 'basic' information. In other words, capture crucial data in databases, which subsequently opens the road for further automation.

Step G, Human aspect: Most important is the human aspect of implementation, which is the most difficult obstacle. From interviews, it became clear that there are a multitude of stakeholders that have several reasons not to use a RDPSM. One example is resource pools who are reluctant to provide even an estimate of their availability, since they feel for example that it may be used to assess their productivity. Furthermore, all planners are already using certain planning and scheduling techniques, so they have to be convinced of the progress of changing. The list goes on. Therefore, a proof of concept is proposed using a small team of willing and innovative employees such that the practical effectiveness is proofed. Such a team is already identified within JIVC based on previous interviews. The results, if positive, can be used to move to the next implementation phase.

6.4.2 Minimal implementation

The RDPSM will not be adopted by a larger user base, since it is just passed a 'proof of concept' phase after completion of this thesis. The following is a description of a minimal implementation plan, which assumes that JIVC is pleased with the potential proven by the previous set of implementation steps, and prepares it to be used by a larger user base. This suggests a professional implementation of the code, which realistically means that software engineers have to reprogram the code. If done in-house, it would require (already) scarce resources at JIVC, which is a huge obstacle. The probability of gaining resources in favour of military strategic projects is negligible. Perhaps more realistically is subcontracting to organisations that are specialized in project planning software. Unfortunately, it is more likely in that scenario that existing project scheduling software is purchased using default scheduling methods that somehow ensure robustness in favour of developing a completely new method. Any step moving towards automated information processing regarding scheduling is probably an improvement compared to the current situation. Regardless of who would do a minimal implementation, the following steps are the same.

Step A, Improving user interface: Currently, the code runs as a Python script, where the MPSPLib instances are located in the same directory and where the output is saved in an output folder. In order to be used in a professional environment, the user interface should be improved, for example by providing a web based GUI. This could enable users to create problem instances inside the web app, to save, load, import and export instances, to interact with the output etc.

Step B, Training: Organize trainings or create tutorials that assist employees to get familiar with the RDPSM and such that they can interpret output results.

Step C, Adjustments of work processes. Work processes could be redesigned to implement a RDPSM as a regular scheduling practice.

Step D, Improve the software from a script to an actual application.

6.4.3 Full implementation

In a full implementation, the RDPSM is used for (automated) scheduling in PM and CM at JIVC. The following steps explain that process.

Step 1: Redesign work processes for required data selection. First of all, project and resource data must be available in databases at JIVC. Data that is of yet missing is workload estimation, resource availability, and relations between tasks, as explained in (subsection 3.4.1). Gathering this data requires back-end development, API's, such that databases can handle the required data and connections to the front-end development such as the GUI mentioned in the previous set of steps. Moreover, it requires employees to cooperate and continuously and reliably submit this data in databases. An implementation program to change the work processes could take a substantial amount of time. It would impact employees involved in CM and PM, in planning, and in decision making such as managers.

Step 2 Improvement: Refine and improve the current scheduling algorithm, for example using the improvement suggestions (see section 5.6). If data is accessible in real-time, rescheduling techniques can be introduced.

6.5 Conclusion RQ7

To conclude this chapter, a summary is given of the answers to the research question.

RQ7: The designed RDPSM is validated. Results indicate that the RDPSM increases robustness as intended, although not as effective for all problem instances, being related to the particular instance characteristics. The effectiveness degrades for more constrained problem instances. Yet, robustness is improved without violating private information in the decentralized setting.

Chapter 7

Discussion and conclusion

The previous chapter evaluated the RDPSM and presented results. This chapter first discusses the results attained in the previous chapter (section 7.1). Subsequently, the conclusion of this thesis is presented (section 7.2).

7.1 Discussion

This section presents the interpretation of the results (subsection 7.1.1), the implication on other scientific research (subsection 7.1.2), the limitations of the research (subsection 7.1.3) and management directions for JIVC (subsection 7.1.4).

7.1.1 Interpretation results

This section presents an interpretation of the results attained in section 6.3.

First and foremost, the results indicate that the RDPSM achieves its main goal: improving robustness of schedules in a multi-project environment while only using private information. It indicates that integrating decentralized project scheduling problems and robust project scheduling problems is feasible.

Suppose JIVC would only implement the replication of the decentralized scheduling method of Song et al. (2016) to generate schedules and use them as internal due dates. If these are subjected to the uncertainty as generated in the simulation, on average 30,4% of the projects are delivered in time (a), meaning that due dates are postponed in the remaining 69,6% of cases. This is reduced to postponing 47,4% when using the RDPSM.

The results also indicate that the RDPSM does not guarantee reliability, as deadlines are still postponed in on average 47,4% of the instances. Yet, this depends heavily on the characteristics of the problem instance such as the constrainedness of resources and use of resources. Compare the eight easier non-AC sets to the four more difficult AC sets in Figure 6.4. This is in line with findings in virtually all other project scheduling experiments in literature, for example, it is also described by Song et al. (2016). Performance is generally degraded for more difficult instances. A positive aspect is that the constrainedness can be measured and evaluated based on the project instance characteristics before starting

the method. Therefore, if new instances are presented, the characteristics can serve as an estimator for the effectiveness of the RDPSM.

An alternative explanation for the results is that the fact of using a later due date is the only cause of in-time delivery improvement. Following that reasoning, only estimating a valid due date would improve in-time delivery. While the later due date is indeed a cause of in-time delivery improvement, it would not help JIVC in preventing due dates to be postponed. First of all, determining such a due date is not evident. Subsequently, finding an accompanying feasible schedule is crucial for execution and schedule repair, since DMUs cannot act without knowledge of the schedule. Yet, an additional experiment comparing the in-time delivery performance between the replicated version that uses an estimated due date and the RDPSM would give more insights in this aspect.

An unexpected result is that the 30_20 instance set has exceptionally low performance. Yet, looking at the characteristics, it is noticed that four of the five instances in the set have high overload factors (or constrainedness). This is most likely the cause of poor performance.

7.1.2 Implication and relation to research

This section elaborates on the implications of the research compared to other research, studies and theories.

In general, the general dynamics of robustness on project scheduling are well known in literature. The results of this thesis regarding robustness are in line compared to other publications that report on robustness, such as W. Herroelen and Leus (2005), W. Herroelen and Leus (2007), Rasconi et al. (2008), Billaut et al. (2008), Schwindt and Zimmermann (2015), and Hazır and Ulusoy (2020). The new component of this thesis regarding robustness is the focus on the integration of a robust method in a decentralized methods. The new problem dynamic introduced is in determining the amount of resources to reserve when adding slack (see subsection 5.3.1), which is a trade-off problem. The implication of this thesis in practice and for the field of research is the consideration of additional research that considers this dynamic.

Besides that, there is a debate regarding the addition of slack in planning and scheduling that deserves attention. The first phenomenon is Parkinson's law (Parkinson, 1958), which claims that when people are given more time to complete work, they will take that time for granted and consume it. This reasoning therefore discourages adding slack. Moreover, the students syndrome phenomenon states that people tend to work harder closer to deadlines. Hence, people tend to start late due to a lack of initial "necessity", which later changes due to increases in stress. This reasoning discourages the use of slack since it postpones a deadline. Goldratt (1997) claims that CCM project management prevents the negative effects of student syndrome and Parkinson's Law but this is disputed. The aspects of these phenomenons are regularly linked in especially IT-development and their effects are researched, as for example by D. C. Smith (2010). These human and psychological aspects in the work that is scheduled are not taken into consideration in the mathematical model of the RDPSM.

Another debate questions the productivity of planning. Kahneman and Tversky (1977) report a bias towards optimistic estimations in planning. This widely disputed phenomenon is named the 'planning fallacy'. A recent study by Zwikael and Gilchrist (2021) broadens

the question and asks when planning is counterproductive. Using risk as a main factor, they conclude that planning may be counterproductive when low risk tasks undergo (what they call) 'tactical' planning. It is a valid question. Using their definition of risk, it seems that the situation at JIVC does not fall into this category.

A special variant of PSPs is the Software Project Scheduling Problem (SPSP). This variant focuses especially on software projects, where Alba and Francisco Chicano (2007) clearly states differences between the SPSP and the RCPSP. They claim that because of these differences, SPSP instances requires another approach and solution method. The reader may argue that these strategies should also be used in this thesis. However, based on the literature in the extensive review on SPSP by Vega-Velázquez et al. (2018), it is concluded that the SPSP does not focus on multi-project instances. The focus is mainly on small problems and instances, often of just a single project. There is uncertainty in using this method multi-project scheduling and larger instances, which is a risk deemed to high for this thesis.

7.1.3 Limitations

The results are limited due to the limited available time and resources available within the scope of the thesis. Some of the limitations are inherent to (project scheduling) problems subject to uncertainty.

Instances

At the initiation of the thesis, the expectation was to conduct a case study using the data at JIVC. Unfortunately, the data was not suitable. Therefore, the RDPSM is tested using the problem instances from the MPSPLib dataset. The characteristics of this dataset are limited as investigated by Eynde and Vanhoucke (2020). Therefore, the performance may deviate from the actual situation.

Scenario generator

The disruptions and disruption scenarios are generated by a generator that only operates within certain parameter limits. Therefore, the results are only valid for the uncertainty that the instances were subjected to. Results are less reliable when values exceed the parameter settings.

Schedule repair

The repair of a schedule in testing was non-optimal and may produce pessimistic results.

Comparison against replication

No comparison is made between the current scheduling practices at JIVC and the RDPSM. The results show the improvement compared to the replication of Song et al. (2016). The current scheduling strategy at JIVC is difficult to meet since not all data is known to recreate scenarios and compare outcomes.

7.1.4 Management directions

This section provides direction to management including advice on how to improve current scheduling practices at JIVC.

There are many manual operations in the current work processes of some employee types at JIVC that are particularly suited to be automated. For example, at least an estimation of the availability of an employee should be readily available in a system.

Overall, it is advised to MinDef to construct data models in their software suites to store valuable data in a natural way. The latter meaning that data could be collected with only minor changes to current work processes, such that employees quickly recognize benefits of automating some processes. To reuse the previous example, only being able to quickly look up an employees (estimated) availability would greatly reduce time spend by planners in the current processes. Capacity management could for example be improved by taking small orders into account. An estimation of the required workload could be made based on historical data. Additional value could be retrieved by determining the throughput, service and waiting times of changes and using statistical methods and modelling on them. Actual performance could be determined, bottlenecks could be identified leading to managerial insights that may improve performance. Examples are relations between customer types, order types, resource pools, sections, etc.

Looking towards the future, as more data is available, schedules could be generated based on readily available data. Moreover, if disruptions and tasks completion is registered in (near) real time, reactive scheduling measures become available.

Another approach to improve the situation is the reduction of the uncertainty and variability. Ward and Chapman (2003) advocate managers to first find the root cause of uncertainty before attempting to control uncertainty. Through this study, some sources have been identified. Yet it is unknown if these are the actual root causes within the complex hierarchical and bureaucratic organisation of JIVC.

7.2 Conclusion

This section concludes the thesis. First, the conclusion is presented (subsection 7.2.1), which is followed by the scientific contribution (subsection 7.2.2) and finished by stating options for further research (subsection 7.2.3).

7.2.1 Conclusion

The research objective is to develop a robust & decentralized project scheduling method to solve a RCMPSP subject to uncertainty and asymmetrically distributed information to increase in-time delivery while minimizing the makespan. This thesis accomplished to design such a RDPSM that deals with both types of information. As a core framework, the publication by Song et al. (2016) is replicated which features a MAS employing an iterative CA. This framework deals with the decentralized aspect and optimizes resource allocation. Improving robustness is not self-evident, the methods that account for the important multi-project aspect cannot be utilized since they violate private information. On the other hand, using single-project robustness methods are not efficient in multi-project environments. The most suitable measures is identified as the STC as described

by Van de Vonder et al. (2008), yet it had disadvantages. Therefore, a custom buffer insertion algorithm is designed based on the STC approach, that employs the *CSS* surrogate robustness measures by Khemakhem and Chtourou (2013). An experiment based on Monte Carlo simulation is conducted to evaluate the RDPSM using MPSPLib as problem instances. The non-robust replication is compared to that which was modified for robustness improvement. On average, the replication delivered 30,45% of instances in time, whereas the RDPSM achieved this for 52,68%. Therefore, the schedules generated by the RDPSM are 22,23% more reliable compared to the non-robust schedules, which is an improvement of 72,99%. It is a trade-off problem, meaning that makespan generally degrades due to improved robustness. On average, the total makespan increased on average by 14,96 days, which is a % 5,42 increase of the makespan. As is generally known in scheduling methods, some instance characteristics such as resource constrainedness have a significant impact on the effectiveness of the method. For some instances, the in-time delivery only increases by 0,80% whereas others increase by 46,52%. Nonetheless, the method improves robustness but does not guarantee in-time delivery. However, it is a trade-off problem, such that parameter configuration may enhance the in-time delivery performance at a further cost of makespan.

The design objective is to improve the in-time delivery and makespan of the current non-optimal scheduling strategy of JIVC. A disruption scenario generator is developed for the experiment that evaluates the performance of RDPSM. This generator replicates the uncertainty as management of JIVC expects it, which is a 10% delay of 10% of all projects in the multi-project instance. As indicated in the previous paragraph, reliability is improved to some degree by adding robustness without violating information privacy. The resulting percentage may not be sufficient for JIVC but this can be configured in parameter settings. The results presented here are constrained due to the limited time and resources available for this thesis.

7.2.2 Contribution

The thesis contributes knowledge to the domain of project scheduling problems. A robust & decentralized project scheduling method does not exist yet to the best knowledge of the author (as explored in Martens (2021)). The reasons why it is not self-evident that these can be combined are violation of private information on one hand and inefficient reservation of slack on the other hand. Hence, the RDPSM contributes to the field (see subsection 5.3.1). A part of the contribution is the problem dynamic of the amount of resources to reserve when adding slack, which is a trade-off.

7.2.3 Further research

This research could be continued upon in the following ways.

Case study

A case study using actual problem instances as was intended reveals the performance of the algorithm in practical situations. Further research that conducts such as study could reveal that performance. In it, the measurement or selection criteria of the winner determination problem could be adjusted. A suggestion is to select the project having

the highest gain in strategic value given the costs of that project. For costs, consider the demand, utilization of a time slot and actual costs/hour of a resource.

Performance of characteristic

The robustness improvement algorithm based on the CSS as devised in this thesis could be evaluated under different circumstances outside a decentralized method such that its individual contribution is determined. Further research could isolate that algorithm and conduct additional experiments.

Amount of resources to add in slack

The problem dynamic in improving robustness, being the trade-off decision of the amount of resources to select in adding slack (see subsection 5.3.1), can be studied in more detail. Further research might investigate the effects under different circumstances and possibly advice in favour or against using certain amounts of slack in certain conditions.

Broader exploration

The exploration within this thesis is limited on a couple of fronts. Further research could widen the scope and implement a broader exploration. For example, elements of SPSP may be explored. Furthermore, the SLR search for a DRCMPSP was very explicit and narrow and some core frameworks might have been overlooked. Furthermore, other MAS methods could be explored such as two-sided auctions (exchanges). Nanda et al. (2017) present various nature-inspired MAS algorithms that solve the RCPSP, claiming that these nature-inspired algorithms are good in handling uncertainty as encountered in reality.

Abstraction of agents

The abstraction of agents may be overhauled to include other decision making aspects. Information may be lost in the current abstraction of multiple DMUs as agent(s) in a MAS. Multiple entities at the organisation are represented by one agent, which is a risk. For example, if each project is solely seen as a separate project agent, information is lost regarding customers. Customers manage a portfolio of projects and can decide how they want to prioritize their own projects and choose which it wants to submit as orders. A MAS however is versatile enough to cope with this problem. Agents may be decoupled and additional abstraction layers may be added to represent DMUs, for example customer agents that prioritize projects and choose which are to be submitted for bidding.

Method improvement

Further research could be devoted to improving the RDPSM. One is the inclusion of reactive scheduling methods, another is the reservation of slack. A third and final is improving computational efficiency.

The inclusion of reactive scheduling methods most likely improves scheduling at MinDef. Van de Vonder, Ballestín, et al. (2007) researched reactive project scheduling methods that repair disrupted baseline schedules. They found a parallel SGS using 'earliest baseline activity starting time' as priority rule to be best performing SGS that repairs a schedule.

More advanced algorithms are priority rule sampling and time window sampling, the latter of which increases computational time. Another approach is the a minimum perturbation strategy W. Herroelen and Leus (2007). Additional attention is required in selecting rescheduling methods, because some may only improve stability robustness while focusing less on quality robustness. For example, rescheduling can be modelled as weighted earliness-tardiness problem, in other words, a weight (or cost) is associated by deviating from the scheduled time E. Demeulemeester and Herroelen (2010). This preserves stability.

An idea is for projects to share reserved slack. Suppose a project is planned an that project uses slack. Slack is known to other projects (similar to the demand ratio) and the project itself also wants to use slack. What if the project is allowed to also use the slack of other projects for itself, such that timeslots with slack get aligned? This would be a 'risk sharing' policy.

Additionally, efficiency of the computational and programming aspects of the RDPSM, but this does not necessarily require research. All but one of these aspects are mentioned in section 5.6. The last aspect is improving the replication of the method. As indicated by the results, the replication performs worse than that reported in the publication. Improving that may also improve the method overall.

Bibliography

Abrache, J., Crainic, T. G., Gendreau, M., & Rekik, M. (2007). Combinatorial auctions. *Annals of Operations Research 2007 153:1*, *153*(1), 131–164. https://doi.org/10.1007/S10479-007-0179-Z

Adhau, S., Mittal, M., & Mittal, A. (2012). A multi-agent system for distributed multi-project scheduling: An auction-based negotiation approach. *Engineering Applications of Artificial Intelligence*, *25*(8), 1738–1751. https://doi.org/10.1016/j.engappai.2011.12.003

Ahmadpour, S., & Ghezavati, V. (2019). Modeling and solving multi-skilled resource-constrained project scheduling problem with calendars in fuzzy condition. *Journal of Industrial Engineering International*, *15*, 179–197. https://doi.org/10.1007/s40092-019-00328-w

Alba, E., & Francisco Chicano, J. (2007). Software project management with GAs. *Information Sciences*, *177*(11), 2380–2401. https://doi.org/10.1016/J.INS.2006.12.020

Alipouri, Y., Sebt, M., Ardeshir, A., & Zarandi, M. (2020). A mixed-integer linear programming model for solving fuzzy stochastic resource constrained project scheduling problem. *Operational Research*, *20*(1), 197–217. https://doi.org/10.1007/s12351-017-0321-x

Ammar, M., & Abd-ElKhalek, S. (2019). Criticality measurement in fuzzy project scheduling. *International Journal of Construction Management*. https://doi.org/10.1080/15623599.2019.1619226

Aramesh, S., Mousavi, S., & Mohagheghi, V. (2021). A new comprehensive project scheduling, monitoring, and management framework based on the critical chain under interval type-2 fuzzy uncertainty. *Iranian Journal of Fuzzy Systems*, *18*(1), 151–170. https://doi.org/10.22111/ijfs.2021.5880

Arık, O. (2019). Credibility based chance constrained programming for project scheduling with fuzzy activity durations. *International Journal of Optimization and Control: Theories and Applications*, *9*(2), 208–215. https://doi.org/10.11121/ijocta.01.2019.00631

Artigues, C., Demassey, S., Néon, E., & Sourd, F. (2010). Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications. *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications*, 1–308. https://doi.org/10.1002/9780470611227

Artigues, C., & Roubellat, F. (2000). A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes. *European Journal of Operational Research*, *127*(2), 297–316. https://doi.org/10.1016/S0377-2217(99)00496-8

Azimi, F., & Fathallahi, F. (2016). Fuzzy multi objective project scheduling under inflationary conditions. *Economic Computation and Economic Cybernetics Studies and Research*, *50*(3), 337–350.

Ballestín, F., & Blanco, R. (2011). Theoretical and practical fundamentals for multi-objective optimisation in resource-constrained project scheduling problems. *Computers & Operations Research*, *38*(1), 51–62. https://doi.org/10.1016/J.COR.2010.02.004

Bhaskar, T., Pal, M. N., & Pal, A. K. (2011). A heuristic method for RCPSP with fuzzy activity times. *European Journal of Operational Research*, *208*(1), 57–66. https://doi.org/10.1016/J.EJOR.2010.07.021

Billaut, J.-C., Moukrim, A., & Sanlaville, E. (2008). *Flexibility and robustness in scheduling*. ISTE/John Wiley & Sons. https://doi.org/10.1002/9780470611432

Birjandi, A., & Mousavi, S. (2019). Fuzzy resource-constrained project scheduling with multiple routes: A heuristic solution. *Automation in Construction*, *100*, 84–102. https://doi.org/10.1016/j.autcon.2018.11.029

Blazewicz, J., Lenstra, J. K., & Kan, A. H. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, *5*(1), 11–24. https://doi.org/10.1016/0166-218X(83)90012-4

Bonnal, P., Gourc, D., & Lacoste, G. (2004). Where Do We Stand with Fuzzy Project Scheduling? *Journal of Construction Engineering and Management*, *130*(1), 114–123. https://doi.org/10.1061/(ASCE)0733-9364(2004)130:1(114)

Browning, T. R., & Yassine, A. A. (2009). A random generator of resource-constrained multi-project network problems. *Journal of Scheduling 2009 13:2*, *13*(2), 143–161. https://doi.org/10.1007/S10951-009-0131-Y

Browning, T. R., & Yassine, A. A. (2010). Resource-constrained multi-project scheduling: Priority rule performance revisited. *International Journal of Production Economics*, *126*(2), 212–228. https://doi.org/10.1016/J.IJPE.2010.03.009

Brucker, P., Drexl, A., Möhring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, *112*(1), 3–41. https://doi.org/10.1016/S0377-2217(98)00204-5

Brucker, P., & Knust, S. (2012). Scheduling Models. *Complex Scheduling*, 1–28. https://doi.org/10.1007/978-3-642-23929-8{_}1

Chaari, T., Chaabane, S., Aissani, N., & Trentesaux, D. (2014). Scheduling under uncertainty: Survey and research directions. *2014 International Conference on Advanced Logistics and Transport, ICALT 2014*, 229–234. https://doi.org/10.1109/ICADLT.2014.6866316

Chanas, S., & Kamburowski, J. (1981). The use of fuzzy variables in pert. *Fuzzy Sets and Systems*, *5*(1), 11–19. https://doi.org/10.1016/0165-0114(81)90030-0

Chen, L., & Zhang, Z. (2016). Preemption resource-constrained project scheduling problems with fuzzy random duration and resource availabilities. *Journal of Industrial and Production Engineering*, *33*(6), 373–382. https://doi.org/10.1080/21681015.2016.1140089

Chen, Z., Demeulemeester, E., Bai, S., & Guo, Y. (2018). Efficient priority rules for the stochastic resource-constrained project scheduling problem. *European Journal of Operational Research*, *270*(3), 957–967. https://doi.org/10.1016/J.EJOR.2018.04.025

Cheng, C.-B., & Lo, C.-Y. (2017). Multi-project scheduling by fuzzy combinatorial auction. *2017 3rd IEEE International Conference on Cybernetics, CYBCONF 2017 - Proceedings*. https://doi.org/10.1109/CYBConf.2017.7985794

Cheng, C.-B., Lo, C.-Y., & Chu, C.-P. (2019). Solving multi-mode resource-constrained multi-project scheduling problem with combinatorial auction mechanisms. *International Journal of Information and Management Sciences*, *30*(2), 143–167. https://doi.org/10.6186/IJIMS.201906{_}30(2).0004

Chtourou, H., & Haouari, M. (2008). A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling. *Computers & Industrial Engineering*, *55*(1), 183–194. https://doi.org/10.1016/J.CIE.2007.11.017

Ciupe, A., Meza, S., & Orza, B. (2016). Heuristic optimization for the resource constrained Project Scheduling Problem: A systematic mapping. *Proceedings of the 2016 Federated Conference on Computer Science and Information Systems, FedCSIS 2016*, 619–626. https://doi.org/10.15439/2016F389

Coelho, J., & Vanhoucke, M. (2020). Going to the core of hard resource-constrained project scheduling instances. *Computers & Operations Research*, *121*, 104976. https://doi.org/10.1016/J.COR.2020.104976

Confessore, G., Giordani, S., & Rismondo, S. (2006). A market-based multi-agent system model for decentralized multi-project scheduling. *Annals of Operations Research 2006 150:1*, *150*(1), 115–135. https://doi.org/10.1007/S10479-006-0158-9

Cramton, P., Shoham, Y., & Steinberg, R. (2005). *Combinatorial Auctions*. The MIT Press. https://doi.org/10.7551/MITPRESS/9780262033428.001.0001

Csébfalvi, A. (2012). A Theoretically Correct Resource Usage Visualization for the Resource- Constrained Project Scheduling Problem. *International Journal of Optimization in Civil Engineering*, 173–181. https://www.academia.edu/2356669/A_Theoretically_Correct_Resource_Usage_Visualization_for_the_Resource_Constrained_Project_Scheduling_Problem

Davonport, A., & Beck, J. (2000). *A survey of techniques for scheduling with uncertainty* (tech. rep.). University of Toronto. Toronto.

Dayoub, N., Stashevskiy, A., Elroba, S., & Elshahate, M. (2020). Application of Fuzzy theory in project scheduling. *Journal of Physics: Conference Series*, *1687*(1). https://doi.org/10.1088/1742-6596/1687/1/012011

De Vries, S., & Vohra, R. V. (2003). Combinatorial auctions: A survey. *INFORMS Journal on Computing*, *15*(3), 284–309. https://doi.org/10.1287/IJOC.15.3.284.16077

Deblaere, F., Demeulemeester, E., & Herroelen, W. (2011). RESCON: Educational project scheduling software. *Computer Applications in Engineering Education*, *19*(2), 327–336. https://doi.org/10.1002/CAE.20314

Deblaere, F., Demeulemeester, E., Herroelen, W., & Vonder, S. V. d. (2007). Robust Resource Allocation Decisions in Resource-Constrained Projects*. *Decision Sciences*, *38*(1), 5–37. https://doi.org/10.1111/J.1540-5915.2007.00147.X

Demeulemeester, E., Deblaere, F., Herbots, J., Lambrechts, O., & Van de Vonder, S. (2007). A Multi-level Approach to Project Management under Uncertainty. *Review of Business and Economic Literature*, *52*(3), 391–409. https://lirias.kuleuven.be/retrieve/110932

Demeulemeester, E., & Herroelen, W. (2010). Robust Project Scheduling. *Foundations and Trends® in Technology, Information and Operations Management*, *3*(3–4), 201–376. https://doi.org/10.1561/0200000021

Demeulemeester, E. L., & Herroelen, W. S. (2002). Classification of Project Scheduling Problems. In F. S. Hillier (Ed.), *Project scheduling* (pp. 71–93). Kluwer Academic Publishers. https://doi.org/10.1007/b101924

d'Inverno, M., & Luck, M. (2004). The Agent Landscape. *Understanding agent systems* (2nd ed., pp. 1–13). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-10702-7{_}1

Elizabeth, S., & Sujatha, L. (2015). Project scheduling method using triangular intuitionistic fuzzy numbers and triangular fuzzy numbers. *Applied Mathematical Sciences*, *9*(1-4), 185–198. https://doi.org/10.12988/ams.2015.410852

Eynde, R. V., & Vanhoucke, M. (2020). Resource-constrained multi-project scheduling: benchmark datasets and decoupled scheduling. *Journal of Scheduling 2020 23:3*, *23*(3), 301–325. https://doi.org/10.1007/S10951-020-00651-W

Fathallahi, F., & Najafi, A. (2016). A hybrid genetic algorithm to maximize net present value of project cash flows in resource-constrained project scheduling problem with fuzzy parameters. *Scientia Iranica*, *23*(4), 1893–1903. https://doi.org/10.24200/sci.2016.3935

Fatima, S., Kraus, S., & Wooldridge, M. (2014). *Principles of Automated Negotiation*. Cambridge University Press. https://doi.org/10.1017/CBO9780511751691

Fink, A., & Homberger, J. (2015). Decentralized Multi-Project Scheduling. *Handbook on Project Management and Scheduling Vol. 2*, 685–706. https://doi.org/10.1007/978-3-319-05915-0{_}2

Fox, G. E., Baker, N. R., & Bryant, J. L. (1984). Economic Models for R and D Project Selection in the Presence of Project Interactions. *http://dx.doi.org/10.1287/mnsc.30.7.890*, *30*(7), 890–902. https://doi.org/10.1287/MNSC.30.7.890

Franklin, S., & Graesser, A. (1996). Is It an agent, or just a program?: A taxonomy for autonomous agents. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1193*, 21–35. https://doi.org/10.1007/BFB0013570

Gazdik, I. (1983). FUZZY-NETWORK PLANNING - FNET. *IEEE Transactions on Reliability*, *R-32*(3), 304–313. https://doi.org/10.1109/TR.1983.5221657

Ghaffari, M., & Emsley, M. W. (2015). Current status and future potential of the research on Critical Chain Project Management. *Surveys in Operations Research and Management Science*, *20*(2), 43–54. https://doi.org/10.1016/J.SORMS.2015.10.001

Goldratt, E. M. (1997). *Critical chain*. North River Press.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. H. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. *Annals of Discrete Mathematics*, *5*(100), 287–326. https://doi.org/10.1016/S0167-5060(08)70356-X

Habibi, F., Barzinpour, F., & Sadjadi, S. J. (2018). Resource-constrained project scheduling problem: review of past and recent developments. *Journal of Project Management*, *3*, 55–88. https://doi.org/10.5267/j.jpm.2018.1.005

Hans, E. W., Herroelen, W., Leus, R., & Wullink, G. (2007). A hierarchical approach to multi-project planning under uncertainty. *Omega*, *35*(5), 563–577. https://doi.org/10.1016/J.OMEGA.2005.10.004

Hartmann, S., & 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. https://doi.org/10.1016/J.EJOR.2009.11.005

Hartmann, S., & Briskorn, D. (2021). An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*. https://doi.org/10.1016/J.EJOR.2021.05.004

Hazır, Ö., Haouari, M., & Erel, E. (2010). Robust scheduling and robustness measures for the discrete time/cost trade-off problem. *European Journal of Operational Research*, *207*(2), 633–643. https://doi.org/10.1016/J.EJOR.2010.05.046

Hazır, Ö., Haouari, M., & Erel, E. (2015). Robust Optimization for the Discrete Time-Cost Tradeoff Problem with Cost Uncertainty. *Handbook on Project Management and Scheduling Vol. 2*, *2*, 865–874. https://doi.org/10.1007/978-3-319-05915-0{_}9

Hazır, Ö., & Ulusoy, G. (2020). A classification and review of approaches and methods for modeling uncertainty in projects. *International Journal of Production Economics*, *223*, 107522. https://doi.org/10.1016/J.IJPE.2019.107522

Herroelen, W., & Leus, R. (2004). The construction of stable project baseline schedules. *European Journal of Operational Research*, *156*(3), 550–565. https://doi.org/10.1016/S0377-2217(03)00130-9

Herroelen, W., & Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, *165*(2), 289–306. https://doi.org/10.1016/J.EJOR.2004.04.002

Herroelen, W., & Leus, R. (2007). Robust and reactive project scheduling: a review and classification of procedures. *https://doi.org/10.1080/00207540310001638055*, *42*(8), 1599–1620. https://doi.org/10.1080/00207540310001638055

Homberger, J., & Fink, A. (2017). Generic negotiation mechanisms with side payments – Design, analysis and application for decentralized resource-constrained multi-project scheduling problems. *European Journal of Operational Research*, *261*(3), 1001–1012. https://doi.org/10.1016/j.ejor.2017.03.022

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. https://doi.org/10.1111/J.1475-3995.2007.00614.X

Homberger, J., Vullriede, R., Horstmann, J., Lanzl, R., Kistler, S., & Göttlich, T. (2007). Eine Web-Service basierte Architektur für ein Multi-Agenten System zur dezentralen Multi-Projekt Planung. *Operations Research Proceedings 2006*, 541–546. https://doi.org/10.1007/978-3-540-69995-8{_}86

Horling, B., & Lesser, V. (2004). A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, *19*(4), 281–316. https://doi.org/10.1017/S0269888905000317

Hu, W., Wang, H., Peng, C., Wang, H., Liang, H., & Du, B. (2015). An outer–inner fuzzy cellular automata algorithm for dynamic uncertainty multi-project scheduling problem. *Soft Computing*, *19*(8), 2111–2132. https://doi.org/10.1007/s00500-014-1395-5

Huang, M., Yuan, J., & Xiao, J. (2015). An adapted firefly algorithm for product development project scheduling with fuzzy activity duration. *Mathematical Problems in Engineering*, *2015*. https://doi.org/10.1155/2015/973291

Joglekar, N. R., Kulatilaka, N., & Anderson, E. G. (2007). Hierarchical planning under uncertainty: Real options and heuristics. *Handbook of new product development management* (1st ed., pp. 307–330). Routledge. https://doi.org/10.4324/9780080554402-16

Kahneman, D., & Tversky, A. (1977). *Intuitive Prediction: Biases and Corrective Procedures* (tech. rep.). https://apps.dtic.mil/sti/citations/ADA047747

Kaveh, A., Khanzadi, M., & Alipour, M. (2016). Fuzzy resource constraint project scheduling problem using CBO and CSS algorithms. *International Journal of Civil Engineering*, *14*(5), 325–337. https://doi.org/10.1007/s40999-016-0031-4

Khalilzadeh, M., Shakeri, H., Gholami, H., & Amini, L. (2017). A Heuristic Algorithm for Project Scheduling with Fuzzy Parameters. *Procedia Computer Science*, *121*, 63–71. https://doi.org/10.1016/j.procs.2017.11.010

Khemakhem, M. A., & Chtourou, H. (2013). Efficient robustness measures for the resource-constrained project scheduling problem. *International Journal of Industrial and Systems Engineering*, *14*(2), 245–267. https://doi.org/10.1504/IJISE.2013.053738

Klimek, M., & Łebkowski, P. (2011). Resource allocation for robust project scheduling. *Bulletin of the Polish Academy of Sciences: Technical Sciences*, *59*(1), 51–55. https://doi.org/10.2478/V10175-011-0008-Z

Knyazeva, M., Bozhenyuk, A., & Kacprzyk, J. (2018). *Modeling decisions for project scheduling optimization problem based on type-2 fuzzy numbers* (Vol. 11288 LNAI). https://doi.org/10.1007/978-3-030-04491-6{_}27

Knyazeva, M., Bozhenyuk, A., & Rozenberg, I. (2015). Resource-constrained Project Scheduling Approach under Fuzzy Conditions. *Procedia Computer Science*, *77*, 56–64. https://doi.org/10.1016/j.procs.2015.12.359

Kolisch, R., & Padman, R. (2001). An integrated survey of deterministic project scheduling. *Omega*, *29*(3), 249–272. https://doi.org/10.1016/S0305-0483(00)00046-3

Kolisch, R. (1996). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, *90*(2), 320–333. https://doi.org/10.1016/0377-2217(95)00357-6

Kolisch, R., & Sprecher, A. (1996). *PSPLIB - a project scheduling problem library*, Universität Kiel. http://hdl.handle.net/10419/149843

Kolisch, R., Sprecher, A., & Drexl, A. (1995). Characterization and Generation of a General Class of Resource-Constrained Project Scheduling Problems. *Management Science*, *41*(10), 1693–1703. https://doi.org/10.1287/MNSC.41.10.1693

Kouvelis, P., & Yu, G. (1997). *Robust Discrete Optimization and Its Applications* (1st ed., Vol. 14). Springer US. https://doi.org/10.1007/978-1-4757-2620-6

Krzeszowska-Zakrzewska, B. (2015). Fuzzy Pareto Dominance in Multiple Criteria Project Scheduling Problem. *Multiple Criteria Decision Making*, *10*, 93–104. https://www.infona.pl/resource/bwmeta1.element.cejsh-6fb0bc56-7032-428a-a3b3-870acadeb7cc/tab/summary

Kuchta, D., Marchwicka, E., & Schneider, J. (2021). Sustainability-oriented project scheduling based on z-fuzzy numbers for public institutions. *Sustainability (Switzerland)*, *13*(5), 1–17. https://doi.org/10.3390/su13052801

Kurtulus, I., & Davis, E. W. (1982). Multi-Project Scheduling: Categorization of Heuristic Rules Performance. *http://dx.doi.org/10.1287/mnsc.28.2.161*, *28*(2), 161–172. https://doi.org/10.1287/MNSC.28.2.161

Lau, J., Huang, G., Mak, K., & Liang, L. (2005). Distributed project scheduling with information sharing in supply chains: Part i - An agent-based negotiation model. *International Journal of Production Research*, *43*(22), 4813–4838. https://doi.org/10.1080/00207540500066796

Lehmann, D., Müller, R., & Sandholm, T. (2013). The Winner Determination Problem. *Combinatorial Auctions*, 297–318. https://doi.org/10.7551/MITPRESS/9780262033428.003.0013

Li, J. (2009). Mas-based negotiation mechanism for ship multi-project scheduling. *Proceedings of the 2009 International Conference on Machine Learning and Cybernetics*, *3*, 1388–1392. https://doi.org/10.1109/ICMLC.2009.5212288

Liu, D., & Hu, C. (2021). A dynamic critical path method for project scheduling based on a generalised fuzzy similarity. *Journal of the Operational Research Society*, *72*(2), 458–470. https://doi.org/10.1080/01605682.2019.1671150

Long, L. D., & Ohsato, A. (2008). Fuzzy critical chain method for project scheduling under resource constraints and uncertainty. *International Journal of Project Management*, *26*(6), 688–698. https://doi.org/10.1016/J.IJPROMAN.2007.09.012

Mahdavi, A., Shirazi, B., & Rezaeian, J. (2021). Toward a scalable type-2 fuzzy model for resource-constrained project scheduling problem. *Applied Soft Computing*, *100*. https://doi.org/10.1016/j.asoc.2020.106988

Mamdani, E. H., & Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, *7*(1), 1–13. https://doi.org/10.1016/S0020-7373(75)80002-2

Martens, E. (2021). *1ML05 literature study* (tech. rep.). Eindhoven University of Technology. Venlo.

Mehta, S. V., & Uzsoy, R. M. (1998). Predictable scheduling of a job shop subject to breakdowns. *IEEE Transactions on Robotics and Automation*, *14*(3), 365–378. https://doi.org/10.1109/70.678447

Mochón, A., & Sáez, Y. (2015a). Combinatorial Auction Models, 105–120. https://doi.org/10.1007/978-3-319-08813-6{_}8

Mochón, A., & Sáez, Y. (2015b). Introduction to Combinatorial Auctions, 87–103. https://doi.org/10.1007/978-3-319-08813-6{_}7

Moradi, M., Hafezalkotob, A., & Ghezavati, V. (2018). Sustainability in fuzzy resource constraint project scheduling in a cooperative environment under uncertainty: Iran's Chitgar lake case study. *Journal of Intelligent and Fuzzy Systems*, *35*(6), 6255–6267. https://doi.org/10.3233/JIFS-171821

Nanda, P., Patnaik, S., & Patnaik, S. (2017). Multi-Agent Optimization of Resource-Constrained Project Scheduling Problem Using Nature-Inspired Computing. *Modeling and Optimization in Science and Technologies*, *10*, 217–246. https://doi.org/10.1007/978-3-319-50920-4{_}9

Parkinson, C. (1958). *Parkinson's law: or, The pursuit of progress*. John Murray.

Patterson, J. H. (1984). A Comparison of Exact Approaches for Solving the Multiple Constrained Resource, Project Scheduling Problem. *http://dx.doi.org/10.1287/mnsc.30.7.854*, *30*(7), 854–867. https://doi.org/10.1287/MNSC.30.7.854

Pellerin, R., Perrier, N., & Berthaut, F. (2020). A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *280*(2), 395–416. https://doi.org/10.1016/J.EJOR.2019.01.063

Pennypacker, J. S., & Dye, L. D. (2002). Project portfolio management and managing multiple projects: two sides of the same coin? *Managing multiple projects : Planning, scheduling, and allocating resources for competitive advantage* (p. 323). Marcel Dekker.

Policella, N. (2005). Scheduling with uncertainty: A proactive approach using Partial Order Schedules. *AI Communications*, *18*, 165–167.

Policella, N., Cesta, A., Oddi, A., Smith, S. F., Cesta, A., Oddi, ·. A., Oddi, A., & Smith, S. F. (2008). Solve-and-robustify. *Journal of Scheduling 2008 12:3*, *12*(3), 299–314. https://doi.org/10.1007/S10951-008-0091-7

Ponz-Tienda, J., Pellicer, E., Benlloch-Marco, J., & Andrés-Romano, C. (2015). The Fuzzy Project Scheduling Problem with Minimal Generalized Precedence Relations. *Computer-Aided Civil and Infrastructure Engineering*, *30*(11), 872–891. https://doi.org/10.1111/mice.12166

Prade, H. (1979). Using fuzzy set theory in a scheduling problem: A case study. *Fuzzy Sets and Systems*, *2*(2), 153–165. https://doi.org/10.1016/0165-0114(79)90022-8

Pritsker, A. A. B., Waiters, L. J., & Wolfe, P. M. (1969). Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach. *Management Science*, *16*(1), 93–108. https://doi.org/10.1287/MNSC.16.1.93

Rasconi, R., Cesta, A., & Policella, N. (2008). Validating scheduling approaches against executional uncertainty. *Journal of Intelligent Manufacturing 2008 21:1*, *21*(1), 49–64. https://doi.org/10.1007/S10845-008-0172-7

Rezakhani, P. (2020). Hybrid fuzzy-Bayesian decision support tool for dynamic project scheduling and control under uncertainty. *International Journal of Construction Management*. https://doi.org/10.1080/15623599.2020.1828539

Rezakhani, P. (2021). Project scheduling and performance prediction: a fuzzy-Bayesian network approach. *Engineering, Construction and Architectural Management*. https://doi.org/10.1108/ECAM-07-2020-0540

Roghanian, E., Alipour, M., & Rezaei, M. (2018). An improved fuzzy critical chain approach in order to face uncertainty in project scheduling. *International Journal of Construction Management*, *18*(1), 1–13. https://doi.org/10.1080/15623599.2016.1225327

Schwindt, C., & Zimmermann, J. (2015). *Handbook on Project Management and Scheduling Vol.1* (C. Schwindt & J. Zimmermann, Eds.). Springer International Publishing. https://doi.org/10.1007/978-3-319-05443-8

Shoham, Y., & Leyton-Brown, K. (2008). Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations* (pp. 361–376). Cambridge University Press. https://doi.org/10.1017/CBO9780511811654

Słowiński, R., & Hapke, M. (2000). *Scheduling Under Fuzziness* (R. Slowinski & M. Hapke, Eds.; 1st ed., Vol. 37). Physica-Verlag Heidelberg. https://www.springer.com/gp/book/9783790818673

Smith, D. C. (2010). The Effects of Student Syndrome, Stress, and Slack on Information Systems Development Projects. *Issues in Informing Science and Information Technology*, *7*.

Smith, R. G. (1980). The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, *C-29*(12), 1104–1113. https://doi.org/10.1109/TC.1980.1675516

Song, W., Kang, D., Zhang, J., & Xi, H. (2016). Decentralized multi-project scheduling via multi-unit combinatorial auction. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, 836–844. https://dl.acm.org/doi/pdf/10.5555/2936924.2937046

Song, W., Kang, D., Zhang, J., & Xi, H. (2017). A multi-unit combinatorial auction based approach for decentralized multi-project scheduling. *Autonomous Agents and Multi-Agent Systems*, *31*(6), 1548–1577. https://doi.org/10.1007/s10458-017-9370-z

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-15*(1). https://doi.org/10.1109/TSMC.1985.6313399

Tian, M., Liu, R. J., & Zhang, G. J. (2019). Solving the resource-constrained multi-project scheduling problem with an improved critical chain method. *https://doi.org/10.1080/01605682.2019.1609883*, *71*(8), 1243–1258. https://doi.org/10.1080/01605682.2019.1609883

Tran, D.-H., Cheng, M.-Y., & Pham, A.-D. (2016). Using Fuzzy Clustering Chaotic-based Differential Evolution to solve multiple resources leveling in the multiple projects scheduling problem. *Alexandria Engineering Journal*, *55*(2), 1541–1552. https://doi.org/10.1016/j.aej.2016.03.038

Tsai, S.-E. (2019). Project Scheduling with Resource Constraints by Fuzzy Gantt Chart and Genetic Algorithm. *Journal of Aeronautics, Astronautics and Aviation*, *51*(4), 391–402. https://doi.org/10.6125/JoAAA.201912{_}51(4).05

Ulusoy, G., & Hazır, Ö. (2021). *An Introduction to Project Modeling and Planning* (1st ed.). Springer International Publishing. https://doi.org/10.1007/978-3-030-61423-2

Van de Vonder, S., Ballestín, F., Demeulemeester, E., & Herroelen, W. (2007). Heuristic procedures for reactive project scheduling. *Computers & Industrial Engineering*, *52*(1), 11–28. https://doi.org/10.1016/J.CIE.2006.10.002

Van de Vonder, S., Demeulemeester, E., & Herroelen, W. (2007). A classification of predictive-reactive project scheduling procedures. *Journal of Scheduling 2007 10:3*, *10*(3), 195–207. https://doi.org/10.1007/S10951-007-0011-2

Van de Vonder, S., Demeulemeester, E., & Herroelen, W. (2008). Proactive heuristic procedures for robust project scheduling: An experimental analysis. *European Journal of Operational Research*, *189*(3), 723–733. https://doi.org/10.1016/J.EJOR.2006.10.061

Van De Vonder, S., Demeulemeester, E., Herroelen, W., & Leus, R. (2005). The use of buffers in project management: The trade-off between stability and makespan. *International Journal of Production Economics*, *97*(2), 227–240. https://doi.org/10.1016/J.IJPE.2004.08.004

van de Vonder, S. (2006). *Proactive-reactive procedures for robust project scheduling* (Doctoral dissertation). https://lirias.kuleuven.be/retrieve/67018

Vanhoucke, M. (2018). Planning projects with scarce resources: Yesterday, today and tomorrow's research challenges. *Frontiers of Engineering Management*, *0*(0), 133–149. https://doi.org/10.15302/J-FEM-2018088

Vega-Velázquez, M. Á., García-Nájera, A., & Cervantes, H. (2018). A survey on the Software Project Scheduling Problem. *International Journal of Production Economics*, *202*, 145–161. https://doi.org/10.1016/J.IJPE.2018.04.020

Vetschera, R. (2013). Negotiation processes: an integrated perspective. *EURO Journal on Decision Processes 2013 1:1*, *1*(1), 135–164. https://doi.org/10.1007/S40070-013-0006-5

Wang, L., Zhan, D., Nie, L., & Xu, X. (2013). Schema and solutions for decentralised multi-project scheduling problem. *International Journal of Computer Applications in Technology*, *46*(2), 142–154. https://doi.org/10.1504/IJCAT.2013.052295

Ward, S., & Chapman, C. (2003). Transforming project risk management into project uncertainty management. *International Journal of Project Management*, *21*(2), 97–105. https://doi.org/10.1016/S0263-7863(01)00080-1

Wellman, M. P., Walsh, W. E., Wurman, P. R., & MacKie-Mason, J. K. (2001). Auction Protocols for Decentralized Scheduling. *Games and Economic Behavior*, *35*(1-2), 271–303. https://doi.org/10.1006/GAME.2000.0822

Wieringa, R. J. (2014). *Design Science Methodology for Information Systems and Software Engineering* (1st ed.). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-43839-8

Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: theory and practice. *The Knowledge Engineering Review*, *10*(2), 115–152. https://doi.org/10.1017/S0269888900008122

Wu, S. D., Storer, R. H., & Chang, P. C. (1993). One-machine rescheduling heuristics with efficiency and stability as criteria. *Computers & Operations Research*, *20*(1), 1–14. https://doi.org/10.1016/0305-0548(93)90091-V

Xu, J., Ma, Y., & Xu, Z. (2015). A Bilevel Model for Project Scheduling in a Fuzzy Random Environment. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *45*(10), 1322–1335. https://doi.org/10.1109/TSMC.2015.2406863

Yang, H., Yuan, Y., Ye, S., & Lin, L. (2021). Fuzzy Resource Constrained Project Scheduling Optimization with Hybrid Multiobjective Genetic Algorithm. *Proceedings of the 2021 International Conference on Bioinformatics and Intelligent Computing, BIC 2021*, 280–285. https://doi.org/10.1145/3448748.3448793

Yang, S.-M., Liu, Y.-C., & Yen, T.-Y. (2018). Integration of Fuzzy Logic and QFD for Critical Chain in Project Scheduling with Uncertainties. *2018 International Conference on System Science and Engineering, ICSSE 2018*. https://doi.org/10.1109/ICSSE.2018.8520264

Yuan, Y., Ye, S., Lin, L., & Gen, M. (2021). Multi-objective multi-mode resource-constrained project scheduling with fuzzy activity durations in prefabricated building construction. *Computers and Industrial Engineering*, *158*. https://doi.org/10.1016/j.cie.2021.107316

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, *8*(3), 338–353. https://doi.org/10.1016/S0019-9958(65)90241-X

Zarandi, M. H. F., Asl, A. A. S., Sotudian, S., & Castillo, O. (2018). A state of the art review of intelligent scheduling. *Artificial Intelligence Review 2018 53:1*, *53*(1), 501–593. https://doi.org/10.1007/S10462-018-9667-6

Zhang, Q., Zhou, J., Wang, K., & Pantelous, A. (2018). An Effective Solution Approach to Fuzzy Programming with Application to Project Scheduling. *International Journal of Fuzzy Systems*, *20*(8), 2383–2398. https://doi.org/10.1007/s40815-018-0542-z

Zoulfaghari, H., Nematian, J., & Nezhad, A. (2016). A resource-constrained project scheduling problem with fuzzy activity times. *International Journal of Fuzzy System Applications*, *5*(4), 1–15. https://doi.org/10.4018/IJFSA.2016100101

Zwikael, O., & Gilchrist, A. (2021). Planning to Fail: When Is Project Planning Counterproductive? *IEEE Transactions on Engineering Management*, 1–12. https://doi.org/10.1109/TEM.2021.3053585

Appendix A

Classic Project Scheduling Problems

This appendix presents knowledge and definitions surrounding classic PSPs.

Practitioners are confronted with PSPs in fields such as project management, manufacturing, construction, new product development, IT-projects, health care, and cloud computing. They require methods to find solutions to the optimization problem and solve their practical PSP. First of all, the classic PSP is presented which only considers a single project and is a generalization of the well-known job shop scheduling problem (section A.1). In most practical problems, multiple projects are considered, which is an extension of the classic PSP explained next (section A.2). Subsequently, the current state of the field of PSP research is presented (section A.3).

A.1 Classic project scheduling problem

This section introduces the classic PSP, which is the Resource Constrained Project Scheduling Problem (RCPSP) (subsection A.1.1). Each problem instance has certain characteristics, which are presented next (subsection A.1.2). Furthermore, the complexity of PSPs is highlighted (subsection A.1.3). Subsequently, the most common method to solve the RCPSP is presented (subsection A.1.4). Following up are performance measures of solutions to the problem (subsection A.1.5). Finally, this section is closed by presenting extensions and variations of the problem (subsection A.1.6).

A.1.1 Resource constrained project scheduling problem

The RCPSP, as introduced by Pritsker et al. (1969), has a set of tasks $\mathcal{I}$ and a set of resources $\mathcal{K}$. Common synonyms for tasks are activities, jobs or nodes. Each task i has a duration d_i of consecutive time-units that are non-preemptive. Regularly, a time horizon T is defined in which a project is scheduled, starting with the first activity at $t = 0$. Each resource k has a renewable resource level R_k, which is renewed at each epoch in time to this fixed level. At each time-unit of the task duration, task i requires a constant resource amount r_{ik} from resource k. A task may require multiple resources. Furthermore, each task has a set of predecessors $Pred_i$ containing preceding tasks and a set of successors $Succ_i$ containing succeeding tasks. Therefore, the tasks form a network of nodes (tasks)

with arcs (relations). This is a Directed Acyclic Graph (DAG) which can be represented as an Activity-On-Node (AON) diagram.

The challenge of the problem is finding starting times S_i for each task i such that an objective is optimized. The vector of starting times S is named a schedule and is a solution to the problem. A schedule is feasible if and only if it is:

1. time feasible: all precedence constraints are satisfied
2. resource feasible: all resource constraints are satisfied

Each task has a finishing time F_i which is implied by the relation $F_i = S_i + d_i$. It is common to add two two dummy tasks to the problem, becoming the first and last tasks in the network, which have a resource requirement r_{ik} and duration d_i of zero. The starting time of a project and finishing time of a project are equal to that of the dummy start task and finish tasks respectively.

There are a couple of definitions for tasks in a generated schedule. Some tasks may be shifted in time without effecting the start times of other tasks. The time a task can shift is called slack sl_i, and is defined as the minimum of the start times of all successors minus the maximum of the latest finish times of all predecessors (Equation D.2). All the tasks that have zero slack are critical tasks. All critical tasks together form the Critical Path (CP).

Some researchers define a solution as a set of related schedules by making use of slack. They reason that from a given schedule, it is easy to obtain other schedules by considering all combinations of task shifts that can be made with the given slack. Two special configurations in a solution set are those where all tasks are scheduled to the front (earliest start times) and back (latest start times).

There are some additional parameters that can be given to a project, which are used for some methods. A project may have a due date (or deadline) in which the project should be scheduled. A project may have associated costs for resource usage or tasks and a revenue for the completion of tasks or a project.

A.1.2 Problem characteristics

A PSP instance has characteristics. It is useful to know these characteristics, because the performance of a solution method may differ based on the problem instance characteristics Eynde and Vanhoucke (2020). Characteristics are different from performance measures in that they are solution independent and can be calculated a priori.

The Critical Path Duration (CPD) is the duration of a unique schedule where all resource constraints are ignored. Every task is scheduled at the first feasible moment, which is directly after all its predecessors are completed. This is the shortest possible time in which a project can be completed. The CPD should not be confused with the CP, which is the collection of tasks with zero slack.

The Utilisation Factor (UF_k), or 'average UF_k', is a measure for the resource contention (or constrainedness) for each resource k (Equation D.5). It divides the resource requirements by the resource availability during the critical path duration. If the requirements exceed the availability of the critical path, the $UF > 1$, therefore giving an indication of resource contention. The Overload Factor (UF) is the maximum utilization factor of a project

(Equation D.6) and has no resource index since it is project specific. Further insights are obtained via the variance of the σ^2_{UF} (Equation D.7).

The Normalised Average Resource Loading Factor' ($NARLF'$) measures when resources are required during the project compared to half of the makespan assuming the critical path scenario.

The Network Complexity (NC) measures the complexity of a network by comparing the number of precedence relationship (non-redundant arcs) in the network to the theoretical maximum (Equation D.9).

A.1.3 Problem complexity

PSPs are intractable and $\mathcal{NP}$-hard (Blazewicz et al., 1983). Therefore, exact methods can only solve small problem instances in reasonable computational time. Most real-life sized problem instances however cannot be solved within reasonable computational time using exact methods. Therefore, all sorts of heuristics and methods were developed, some overviews are provided for heuristics (Ciupe et al., 2016), hybrid meta-heuristics (Pellerin et al., 2020) and intelligent scheduling methods (Zarandi et al., 2018). The instance sizes of the PSP at hand cannot be solved with exact methods in a reasonable time. Furthermore, all successive PSPs in this thesis are $\mathcal{NP}$-hard.

A.1.4 Solving the problem

In general, there are two approaches for solving the RCPSP, those from a task point-of-view and those from a resource point-of-view.

The majority of project scheduling methods are constructed from a task point-of-view. Of those methods, the most basic and popular one is a Schedule Generation Scheme (SGS), which generates a schedule according to a set of rules. This is a heuristic method that is quick, gives good results and can be modified. First, an activity list is generated, which is a set of eligible tasks that are feasible to be scheduled at that iteration. Next, a priority rule is applied which selects a certain task from the activity list by selecting the minimum or maximum attribute of the eligible tasks. This task is scheduled after which the two steps are repeated until all tasks are processed. An influential work is that of Kolisch (1996), who evaluated the performance of the most common priority rules. An SGSs can be categorized into a serial and parallel category. After schedule generation, improvement methods could be applied, and among those 'backward and forward scheduling' and 'justification' are popular. The SGS is popular because it is easy to implement, has quick execution time and yields good solutions.

The less common approach is to build schedules from a resource point-of-view instead of a task point-of-view. These methods more related to tactical decision making and corresponding topics such as resource leveling.

A.1.5 Objective functions and measurements

The RCPSP is an optimization problem. As such, it requires at least one objective function that is being minimized or maximized. Measurements are required to use as

objective function and to grade solutions, and these require at least a schedule S in order to be determined.

Objective function categories

Many PSP measures and objective functions exists, and the majority can be allocated into three main categories, being time-based measurements, resource-based measurements and multi-objective optimization (Brucker & Knust, 2012).

Time-based measurements are most popular, and within this category, minimizing the Makespan (MS) is most used (Equation D.10).

The makespan is the maximum task finishing time. If dummy tasks are used, this is equal to the start and finish time of the last dummy task. Other objectives functions are for example '(weighted) (total) flow time', 'tardiness', '(total) (weighted) lateness', '(weighted) number of late activities', etc.. Another perspective on resource usage is a net present value objective, approaching it cost oriented. It associates a cash flow for the completion of tasks and attempts to optimize it.

Resource based measurements are less common. Examples are resource leveling, looking at the deviations or overloads and variation. It also has a cost oriented variant, being the resource investment problem.

Finally, there are multi-objective optimizations containing combinations of measurements and objectives, such as those reviewed in Ballestín and Blanco (2011).

Other measures

Following up are a few measures which are not as often used as objective functions but are important to evaluate the quality of a solution. The Project Delay (PD) is the difference between the makespan of a project and the critical path duration, indicating how much longer it takes to complete a project compared to its theoretical shortest duration. The Relative Gap (RG) is the project delay divided by the critical path duration, giving a relative indication as a percentage of how much longer the project duration is compared to its theoretical shortest duration. The Resource Utilization (RU_k) is equal to the UF_k, except for denominator, which is replaced by the resource availability over the actual duration instead of the critical path duration.

A.1.6 Variations and extensions

The classic RCPSP is fairly basic and has restricting assumptions, hence it is not applicable to most problems in practice. Therefore, researchers extended the model of Pritsker et al. (1969). In the subsequent decades, new generalizations, alternative constraints, other objective functions, and novel solution methods were proposed. Examples are for example those dealing with time-dependent resource profiles, multi-mode cases, non-renewable or partially renewable resources. One of the most relevant project scheduling research directions focused on the consideration of multiple projects which is presented in the next section. Problems related to the RCPSP are the Rough Cut Capacity Planning (RCCP) at a tactical decision making level, and the 'project portfolio selection' problem (Fox et al., 1984) at a strategical decision making level, neither is in the scope of this thesis.

A.2 Classic multi-project scheduling problem

The problem of scheduling multiple projects is known as the Resource Constrained Multi-Project Scheduling Problem (RCMPSP). The collection of multiple projects is regularly referred to as a (project) portfolio. The basic model is presented first (subsection A.2.1). Next, RCMPSP instance characteristics are presented (subsection A.2.2). Subsequently, the most common methods too obtain a solution are explained (subsection A.2.3). Finally, measurements to assess the quality of a solution are shown (subsection A.2.4).

A.2.1 Resource constrained multi-project scheduling problem

The RCMPSP consists of the set $\mathcal{J}$ of projects using index j for each project. Next, the model is equal to that of the RCPSP model, only adding an index j to most parameters such as $\mathcal{I}_j$, d_{ij}, r_{ijk}, d_{ij}, S, etc.

The introduction of multiple projects introduces new challenges. There is now a situation in which resources are shared between projects, named global resources. Some RCMPSP instances only consider global resources whereas other also include local resources which can only be used by one project.

Single-project characteristics and measurements must be adjusted to function in a multi-project setting. In single project scheduling, the first activity is scheduled at $t = 0$. In the new setting, a project may start at a time other than 0 because a lack of available resources. This impacts the formulas previously presented. For example, the makespan of a project was defined as the latest finishing time. This is inaccurate if the project starts at $t > 0$. Therefore, the makespan is now calculated as the difference between the latest ending task and first starting task, since the task may not start at $t = 0$.

Additionally, an earliest start date *esd* (or arrival date, release date) is introduced, indicating that a project may only be scheduled from that particular epoch onward.

A.2.2 Problem characteristics

Characteristics of an RCMPSP are introduced.

The definition of the CPD is extended to multi-project instances. Each single-project receives a project index j and the The Portfolio Critical Path Duration (CPD_{max}) is the maximum CPD of all projects (Equation D.14). The concept of the UF_k is equal to that of single-project instances except that an extra summation is applied for global resources that aggregates the total demand for a single resource over all projects j. The same applies to the $NARLF'$ which is also modified such that it iterates over all projects to aggregate resources.

A.2.3 Solving the problem

The earliest approach to solve an RCMPSPs was to encapsulate all projects into one large project by merging the dummy start and finish moments. Afterwards, regular RCPSP methods could be applied, since it now represents a 'single-project' instance. Although working, this approach does not present good solutions. Better solutions can be obtained

by keeping the projects separated and by including project information in the algorithm. Generally, this still involves RCPSP methods being used to generate schedules or partial schedules for a single project instance.

SGSs remain popular. Kurtulus and Davis (1982) presents the performance of the most common priority rules in multi-project situations. Browning and Yassine (2010) present an extensive publication on priority rules in RCMPSP in which they define characteristics of a PSP. They show which priority rules are best suited given the characteristics of a PSP. Eynde and Vanhoucke (2020) extend on the work by reviewing these measures, proposing adaptations for the measures, and rank the priority rules.

A.2.4 Objective functions and measurements

Objective functions and measurements for the RCMPSP are presented next.

The Start Delay (SD) is defined as the difference between the earliest start date and its actual scheduled start. The Total Makespan (TMS) is the difference between the earliest start time of all projects and the latest finishing time of the last project. The Average Makespan (AMS) is, as the name implies, the average of all single project makespan (Equation D.17). The Average Project Delay (APD) compares the makespan to the critical path of a project (Equation D.18). The Standard Deviation of the Project Delay (DPD) is the corresponding standard deviation of the APD. The portfolio delay ($PDEL$) is similar to the project delay but for the whole portfolio (Equation D.20).

A.3 Project scheduling problem research

This section highlights research in PSPs. The field of research regarding PSPs is vast and diverse. Classifications were devised by researchers to distinguish between particular problem types, which are presented first (subsection A.3.1). Next, terms used in PSP are defined. This thesis is limited to topics relevant to the problem, therefore an overview for further reading to other PSP topics is given (subsection A.3.3). This section is closed by a summary on the current state of the field of research (subsection A.3.4).

A.3.1 Classifications

Classification methods were proposed to distinguish between the different scheduling problems. There is no widely accepted unified project scheduling nomenclature, taxonomy, topology or classification, making it difficult to properly distinguish certain PSP types and methods. The most notable classification methods are those by Graham et al. (1979) Brucker et al. (1999) and E. L. Demeulemeester and Herroelen (2002). Furthermore, Schwindt and Zimmermann (2015) presents a classification of problems based on nine aspects (Table A.1).

Table A.1: Classification of project scheduling problems, from Schwindt and Zimmermann (2015)

Attributes	**Characteristics**
Type of constraints	Time-constrained problem Resource-constrained problem
Type of precedence relations	Ordinary precedence relations Generalized precedence relations Feeding precedence relations
Type of resources	Renewable resources Nonrenewable resources Storage resources Continuous resources Partially renewable resources
Type of activity splitting	Non-preemptive problem Integer preemption problem Continuous preemption problem
Number of execution modes	Single-modal problem Multi-modal problem
Number of objectives	Single-criterion problem Multi-criteria problem
Type of objective function	Regular function Non-regular function
Level of information	Deterministic problem Stochastic problem Problem under interval uncertainty Problem under vagueness
Distribution of information	Centralized problem (symmetric distribution) Decentralized problem (asymmetric distribution)

A.3.2 Terms

One recognized issue is the irregular usage of terms in project scheduling, therefore a list of terms is curated in the glossary.

A.3.3 Overview

The complete field of research regarding PSPs is too extensive to cover entirely, therefore only a selection relevant to this thesis is presented in the next two sections. These elaborate on PSPs subject to uncertainty and PSPs subject to asymmetric information respectively. For other research on PSPs, the following sources are recommended; the original survey by Hartmann and Briskorn (2010) and their recently updated version Hartmann and Briskorn (2021) provide an overview of variants and extensions of the RCPSP; the book by Artigues et al. (2010) presents models, algorithms, extensions and applications; the comprehensive *Handbook on Project Management and Scheduling* of Schwindt and Zimmermann (2015) provides a wider overview on project scheduling; furthermore, extensive reviews are those of Kolisch and Padman (2001) and the more recent review of Habibi et al. (2018).

A.3.4 Active field of research

Even though the vast and extensive collection of research in the past fifty years, PSPs is still an active field of research. First of all due to its practical relevance in project management and scheduling. Secondly, because of its challenging nature as an optimization problem to attain good solutions. Finally, there are still many particular practical situations that require generalized or extended models such that methods can be applied in practice. Vanhoucke (2018) collected and present research questions still open in this field.

Appendix B

Information distribution

This appendix presents background information on information distribution mainly based on the publication by Fink and Homberger (2015). First of all, the types of information distribution are introduced (section B.1). Next, a motivation is given to reject the assumption of symmetric information distribution at JIVC (section B.2). Finally, the difference between a decentralized project scheduling and distributed project scheduling is explained (section B.3).

B.1 Types

The two types of information distribution are symmetric (subsection B.1.1). and asymmetric (subsection B.1.2).

B.1.1 Symmetric

Symmetric information distribution is related to the terms centrally, globally or publicly available information. Assuming symmetric information distribution, all information is global knowledge that can be used by a central Decision Making Unit (DMU) such as the management of an organisation or department. Usually, the decision are made to optimize one or more objectives, for example in project scheduling: completing all projects in a portfolio with minimized time, minimized costs and maximized quality. The assumption of symmetric information is often violated in multi-project scheduling situations, especially in larger hierarchical organisations.

B.1.2 Asymmetric

Symmetric information distribution is related to the terms decentral, distributed or private information. Assuming asymmetric situation, information is distributed over multiple decentralized DMUs that have access to some global information, but also possess local (or private) information that is not shared. In the case of project scheduling, these DMUs could be project managers, resource managers or certain departments. The DMUs act by themselves and may all have different objectives, for example, project owners may want to finish their projects by minimizing the project makespan. Given scarce resources, conflicts arise between these DMUs, since they now compete against each other for limited

resources. DMUs may decide to share or communicate particular information to other DMUs, moreover, they may even communicate biased information (lie) in their pursuit to achieve or optimize their own objective. For example, due dates may be pulled forward to artificially create a higher importance to complete it earlier, or resource requirements may be overstated to 'reserve' more capacity than actually required. Asymmetric information should not simply be overlooked or underestimated, as simply disregarding it results in non-optimal resource allocation and non-optimal scheduling.

B.2 Motivation to reject assumption

There are two reasons to reject the assumption of symmetric information distribution at JIVC. First of all, MinDef is an organization where hierarchy is deeply rooted in the organizational structure (Figure 1.1), making it very likely that information is not centrally available but held by separate DMUs. Furthermore, recall that demand for resources greatly exceeds supply, therefore not all orders can be accepted which creates a competitive environment. Hence, DMUs are self interested and may benefit by withholding or changing information. This is ample evidence to reject the assumption of symmetric information. Therefore, taking asymmetric distributed information into account in decision making will likely improve resource allocation.

Furthermore, according to Hans et al. (2007), multi-project planning approaches often regard projects as single-project planning problems were resource allocation is based on priority and status. Hans et al. (2007) indicate that these fail to consider the effects of multi-project planning at an aggregate capacity level, hence, these fail to take advantage of capacity flexibility.

B.3 Decentralized versus distributed

An important distinction in terminology is that between decentralized project scheduling and distributed project scheduling, as described by Fink and Homberger (2015): *"This intrinsic decentralized organizational character of the DRCMPSP, with multiple autonomous decision makers, must be distinguished from technical distribution, i.e., some distributed implementation of a solution procedure that nonetheless is centrally devised. The latter case would allow to design a distributed solution procedure that supposes that all information about multiple projects is honestly disclosed and thus in principal globally available [...], while in a genuine decentralized situation rational decision makers may communicate biased information to influence the project scheduling in their interest (for example, proclaiming overstated lateness penalties to work towards an earlier scheduling of particular project activities)."*. This distinction however is not used by all researchers in the field. An interaction protocol dictates which information is being shared, therefore it should always be inspected to check if a method is truly decentralized.

Appendix C

Decentralized problem scheduling

This appendix presents additional information on decentralized project scheduling and is organized in three parts. The first introduces MASs (section C.1), and the second discusses auctions (section C.2).

C.1 Multi Agent Systems

This section presents back ground information on the MAS. First, a definition of an agent is given (subsection C.1.1). Subsequently, the reasoning for using a MAS for resource allocation is presented (subsection C.1.2). Finally, a motivation is given why a PSP is interchanged for a WDP (subsection C.1.3).

C.1.1 Definition of an agent

A MAS has DMUs that are represented by autonomous agents. As indicated by d'Inverno and Luck (2004), the definition of an agent is obscure due to the numerous definitions used in literature. The definition of an agent as presented by Franklin and Graesser (1996) is used in this thesis, being: *"An **autonomous agent** is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future."*. The definitions of the weak and strong notion of agenthood by Wooldridge and Jennings (1995) is also utilized. The definitions of agents, agent autonomy, MAS-frameworks and MAS-notations are a research topic on its own, which is not within the scope of this thesis. For more information and discussion on the different views, the book by d'Inverno and Luck (2004) is recommended. Furthermore, Shoham and Leyton-Brown (2008) provide an excellent introduction to multi-agent systems.

C.1.2 Using agents for resource allocation

There are a few distinct ways in which DMUs in an actual organization can interact. Horling and Lesser (2004) describe these types of multi-agent organisations, along with characteristics and accompanying advantages and disadvantages (Table C.1).

Table C.1: Comparing the qualities of various organization paradigms, from Horling and Lesser (2004)

Paradigm	Key characteristic	Benefits	Drawbacks
Hierarchy	Decomposition	Maps to many common domains; handles scale well	Potentially brittle; can lead to bottlenecks or delays
Holarchy	Decomposition with autonomy	Exploit autonomy of functional units	Must organize holons; lack of predictable performance
Coalition	Dynamic, goal-directed	Exploit strength in numbers	Short-term benefits may not outweigh organization construction costs
Team	Group level cohesion	Address larger grained problems; task-centric	Increased communication
Congregation	Long-lived, utility-directed	Facilitates agent discovery	Sets may be overly restrictive
Society	Open system	Public services; well-defined conventions	Potentially complex, agents may require additional society-related capabilities
Federation	Middle agents	Matchmaking, brokering, translation services; facilitates dynamic agent pool	Intermediaries become bottlenecks
Market	Competition through pricing	Good at allocation; increased utility through centralization; increased fairness through bidding	Potential for collusion, malicious behavior; allocation decision complexity can be high
Matrix	Multiple managers	Resource sharing; multiply influenced agents	Potential for conflicts; need for increased agent sophistication
Compound	Concurrent organizations	Exploit benefits of several organizational styles	Increased sophistication; drawbacks of several organizational styles

Horling and Lesser (2004) states two disadvantages of market organisations. The first is the setup of the agent interaction, it is not guaranteed that agents behave honest in interactions, introducing game theory as an important component for interactions. The second is the security of the auction, meaning the protection of information in communication in an open system and the verification of the auction validity. To solve these problems, interaction protocols are employed that limits agent interaction and aim

for incentive-compatible behaviour. Horling and Lesser (2004) first disadvantage regarding the setup of the agent interaction reflects on the challenges in setting up a proper bidding language, and choosing a winner after all bids are submitted. The second disadvantage reflects on what information is shared in a bidding language, since that information is no longer private.

C.1.3 Ratio of interchanging problem

One might question why one intractable problem (solving a scheduling problem) is being replaced for another intractable problem (determining a winner). Due to the distributed design, the latter problem is decoupled into subproblems that can be solved approximately. Given a good mechanism design, this is efficient while still yielding good solutions. This underlines the importance of good mechanism design.

C.2 Auctions

This section presents three topics. The first presents different auction types (subsection C.2.1). The second shows some of the concepts of auction design not yet discussed (subsection C.2.2). The last regard literature on combinatorial auctions (subsection C.2.3).

C.2.1 Auction types

There are many types of auctions, the most common being the Dutch (or descending) auction, English (or ascending) auction, Vickrey auction, and Japanese auction. At a high level, most can be distinguished by looking at two main characteristics, being the bidding rule and pricing rule. From these bidding and pricing rules, auctions are designed through different combinations.

The bidding rule determines how bidders are allowed to place bids, all having certain advantages and disadvantages. Examples are single round (or sealed bid) auctions, dynamic auctions in discrete rounds (iterative auction, multi-round auction) and continuous dynamic auctions (clock auction). The second important characteristic is the pricing rule, which determines what a bidder pays after winning a bid. Examples are paying the value of the winning bid or that of the highest losing bid. Generally, auctions are either optimal (maximizes expected revenue of seller) or efficient (goods are allocated to bidders with highest valuation).

C.2.2 Untouched auction concepts

A couple of important concepts related to auctions are introduced. The first is price discovery in dynamic auctions. After the first bid is placed, information is revealed about the valuation of the good. This information that can be used in succeeding rounds and influences the behaviour of auctions. Another concept is price development, indicating how prices progress in dynamic auctions. Following up is 'collusion', which is when multiple bidders work together to manipulate an auction. Some auction types are vulnerable to collusion, which is a risk and should be avoided.

C.2.3 Combinatorial auctions

Wellman et al. (2001) investigated auction protocols for decentralized project scheduling. Due to the applicability of CAs outside of project scheduling, there are many works on CAs. De Vries and Vohra (2003) present a survey on the design of CAs. An extensive, influential and freely available work on combinatorial auctions is that of Cramton et al. (2005). Abrache et al. (2007) present basic guidelines for designing combinatorial auctions, based on the most important issues encountered in that process, such as the WDP, bidding language (Shoham & Leyton-Brown, 2008, see figure 11.4), valuation functions, information revelation and progressive auctions. Furthermore, the book by Shoham and Leyton-Brown (2008) presents a section dedicated to protocols for resource allocation in combinatorial auctions and guide readers in mechanism design. They indicate the main design space in six aspects, being timing issues, information feedback, bidding rules, termination conditions, bidding language and proxy agents. Mochón and Sáez (2015b) present an extended introduction to CAs. Moreover, Mochón and Sáez (2015a) introduce three iterative CA models, being the ascending proxy auction, clock proxy auction, combinatorial clock auction.

Appendix D

Formulas

This appendix contains all formulas. The formulas are divided over four sections, being those on the RCPSP (section D.1), RCMPSP (section D.2), robustness (section D.3) and solution methods (section D.4).

D.1 RCPSP

This section is divided into formulas on tasks (subsection D.1.1), characteristics (subsection D.1.2) and measurements and objectives (subsection D.1.3). In these formulas, there is no index j to indicate that it concerns a particular project since only one project is under consideration. The next section considers multiple projects and may therefore reuse these formulas by adding an index j to some of the variables following.

D.1.1 Tasks

The following formulas are task specific.

Finish time

The finish time is the start time plus the duration.

$$F_i = S_i + d_i \tag{D.1}$$

Slack

The slack is the difference between the latest start time (maximum) of all predecessors and the earliest (minimum) start time of all successors.

$$sl_i = \min_{succ \in Succ_i} S_{succ} - \max_{pred \in Pred_i} F_{pred} \tag{D.2}$$

D.1.2 Characteristics

Descriptions of characteristics of a problem instance.

Project start time

The scheduled time of the earliest starting task.

$$S_{min} = \min_{i \in \mathcal{I}} S_{i,j} \tag{D.3}$$

Project finish time

The finishing time of the latest finish task.

$$F_{max} = \max_{i \in \mathcal{I}} F_{i,j} \tag{D.4}$$

Critical path duration

From http://www.mpsplib.com/glossary.php. The makespan of a schedule with infinite resource availability.

Utilization Factor

As described in Eynde and Vanhoucke (2020), the average utilization factor or modified average utilization factor, originally described as multi-project measure but modified here as single project measure.

$$UF_k = \frac{\sum_{i \in I} r_{ik}}{R_{ik} \cdot CPD} \tag{D.5}$$

The nominator contains the resource requirements whereas the denominator contains the resource availability during the critical path duration.

Overload Factor

As described in Eynde and Vanhoucke (2020), originally described as multi-project measure but modified as single project measure.

$$UF = \max_{k \in \mathcal{K}} UF_k \tag{D.6}$$

Variance of Utilization Factor

As described in Eynde and Vanhoucke (2020), originally described described as multi-project measure but modified as single project measure.

$$\sigma^2_{UF} = \frac{\sum_{k \in \mathcal{K}} (UF - UF_k)^2}{|\mathcal{K}|} \tag{D.7}$$

Normalised Average Resource Loading Factor

As described in Eynde and Vanhoucke (2020), originally described as multi-project measure but modified as single project measure.

$$NARLF' = \frac{1}{CPD} \sum_{t=0}^{CPD} \sum_{i \in \mathcal{I}} \sum_{k \in H_i} Z'_t X_{it} \left(\frac{r_{ik}}{|H_i|} \right) \tag{D.8}$$

$$\text{with } H_i = k \text{ if } r_{ik} > 0 \qquad \forall k \in \mathcal{K}$$

$$\text{with } z'_t = \begin{cases} -1 & \text{if } t \leq \lceil CPD/2 \rceil \\ 1 & \text{if } t > \lceil CPD/2 \rceil \end{cases}$$

$$\text{and with } X_{it} = \begin{cases} 1 & \text{if activity } a_i \text{ is active at time } t \\ 0 & \text{otherwise} \end{cases}$$

Network complexity

As described in Eynde and Vanhoucke (2020), originally described as multi-project measure but modified as single project measure.

$$NC = \frac{A' - A'_{min}}{A'_{max^*} - A'_{min}} \tag{D.9}$$

$$\text{with } A'_{min} = |I| - 1 \text{ and with } A'_{max} = \frac{|\mathcal{I}|^2}{4}$$

Where A' is the number of precedence relationship arcs.

D.1.3 Measurements and objectives

Measurements of a solution after a problem instance is processed by a method.

Makespan

From http://www.mpsplib.com/glossary.php. The makespan is the complete duration of a project.

$$MS = \max_{i \in \mathcal{I}} F_i = F_{max} \tag{D.10}$$

Project Delay

From http://www.mpsplib.com/glossary.php. The difference between the makespan and the critical path duration.

$$PD = MS - CPD \tag{D.11}$$

Relative Gap

From http://www.mpsplib.com/glossary.php. The project delay as percentage of the critical path duration.

$$RG = \frac{MS - CPD}{CPD} \cdot 100 \tag{D.12}$$

Resource Utilization

From http://www.mpsplib.com/glossary.php. Total usage of a resource.

$$RU_k = \sum_{i \in \mathcal{I}} r_{ik} \cdot d_i \tag{D.13}$$

D.2 RCMPSP

This section is subdivided into characteristics (subsection D.2.1) and measurements and objectives (subsection D.2.2). Some of the formulas in the previous section are reused in addition to the index j to indicate a particular project.

D.2.1 Characteristics

Descriptions of characteristics of a multi-project problem instance.

Portfolio critical path duration

From http://www.mpsplib.com/glossary.php. The minimum duration of the whole portfolio, which is equal to the maximum critical path duration of any of its projects.

$$CPD_{max} = \max_{j \in \mathcal{J}} CPD_j \tag{D.14}$$

D.2.2 Measurements and objectives

Measurements of a multi-project solution after a multi-project problem instance is processed by a method.

Start Delay

From http://www.mpsplib.com/glossary.php. The difference between the earliest start date and the actual start date.

$$SD_j = S_{min,j} - esd_j \tag{D.15}$$

Total Makespan

From http://www.mpsplib.com/glossary.php. Difference between the earliest start time of a project portfolio and latest task finish time of a project portfolio.

$$TMS = \min_{j \in \mathcal{J}} esd_j - \max_{j \in \mathcal{J}} F_{max,j} \tag{D.16}$$

Average makespan

From http://www.mpsplib.com/glossary.php. The project makespan averaged over all projects

$$AMS = \frac{\sum_{j \in \mathcal{J}}}{|\mathcal{J}|} \tag{D.17}$$

Average Project Delay

From http://www.mpsplib.com/glossary.php. The project delay averaged over all projects.

$$APD = \frac{\sum_{j \in \mathcal{J}} PD_j}{|\mathcal{J}|} \tag{D.18}$$

The standard deviation of the average project delay.

$$DPD = \sqrt{\frac{\sum_{j \in \mathcal{J}} (PD_j - APD)^2}{|\mathcal{J}| - 1}} \tag{D.19}$$

Portfolio Delay

From Browning and Yassine (2010). The length of the total makespan of the portfolio compared to the maximum critical path duration of the portfolio.

$$PDEL = \frac{TMS - CPD_{max}}{CPD_{max}} \tag{D.20}$$

D.3 Robustness

This section is subdivided into quality (subsection D.3.1) and stability (subsection D.3.2).

D.3.1 Quality

Robustness measures related to quality robustness.

Combined Slack Sufficiency

From Khemakhem and Chtourou (2013). For each task, sum two numbers. The first number is a score of 1 if the task has sufficient slack. The second number is a score of 1 for every anterior task if the task at hand has sufficient slack to absorb that anterior task. Anterior tasks are all tasks finishing before or at the moment of the scheduled start epoch of the task at hand.

$$CSS = \sum_{i \in \mathcal{I}} \beta_i + \left(\sum_{\{x \in \mathcal{I} \setminus \{i\} | Fx \leq S_i\}} \sigma_x \right) \tag{D.21}$$

$$\text{with } \beta_i = \begin{cases} 1 & \text{if } sl_i \geq SP\% \cdot d_i \\ 0 & \text{otherwise} \end{cases}$$

$$\text{and with } \sigma_j = \begin{cases} 1 & \text{if } sl_i \geq SP\% \cdot d_j \\ 0 & \text{otherwise} \end{cases}$$

$$\text{with } SP\% = \text{slack percentage (expected duration increase)}$$

D.3.2 Stability

Robustness measures related to stability.

Sensitivity

From Rasconi et al. (2008). The sensitivity is the difference between the start times of the disrupted schedule and original schedule.

$$SENS = \frac{\sum_{i \in \mathcal{I}} |S_{disrupted,i} - S_{original,i}|}{|\mathcal{I}|} \tag{D.22}$$

D.4 Method performance

Appendix E

Robust scheduling methods

Table E.1: Project scheduling methods, from E. Demeulemeester and Herroelen (2010)

			Proactive		Reactive			
					Single mode		Multi-mode	
	Deter-ministic	Stochastic Time uncertainty	Time uncertainty	Resource uncertainty	Time uncertainty	Resource uncertainty	General disruption types	Time/ resource disruptions
Ignores resources	CPM	PERT						
Considers resources	RCPSP	SRCPSP	Robust resource allocation	No advance information	WET	Exact procedure	Hybrid MIP/CP MIP/CP	Dedicated procedures
			Quality robustness: critical chain	Advance uncertainty information	Sampling	Heuristics		
			Solution robustness: exact and suboptimal procedures					

Appendix F

Exploration of fuzzy scheduling

Fuzzy project scheduling is a method suited for PSPs subject to vagueness. Fuzzy project scheduling is based on fuzzy set theory, which is explained first (section F.1). Subsequently, its application as project scheduling is introduced (section F.2). The expectation is that fuzzy project scheduling is valuable for application in JIVC, therefore an SLR is conducted to find candidate methods (section F.3). Unfortunately, after execution of the SLR, it is concluded that fuzzy project scheduling is unsuitable since it requires multiple inputs that JIVC cannot provide.

F.1 Fuzzy set theory

Fuzzy project scheduling employs fuzzy set theory to deal with problems subject to vagueness. Fuzzy set theory was first defined by Zadeh (1965). Traditionally, an object is either part of a set or it isn't. A fuzzy set defines membership functions, grading to what degree an object is member of a set using a value in the range [0, 1]. Fuzzy set theory is extended to fuzzy logic by including logical operators and conditional statements. Logic operations such as basic addition and subtraction are not evident using fuzzy sets, moreover, multiple definitions exist to apply operations. Subsequently, procedures were defined applying fuzzy logic, which are Fuzzy Inference System (FIS). A FIS generally consists of three consecutive phases, being: 1) fuzzification, 2) inference and 3) defuzzification, were inference consists of applying logic operations, rule implications and result aggregation. The two most used FIS are the Mamdani FIS (Mamdani & Assilian, 1975) and the Takagi-Sugeno FIS (Takagi & Sugeno, 1985).

F.2 Project scheduling

Prade (1979) was the first to apply fuzzy logic to scheduling problems, followed by the works of Chanas and Kamburowski (1981) and Gazdik (1983) in the subsequent years. Most research focused on describing activity duration using trapezoidal or triangular fuzzy membership functions. Słowiński and Hapke (2000) and Bonnal et al. (2004) published an extensive review on fuzzy project scheduling, whereas W. Herroelen and Leus (2005) published a broader review on uncertainty in project scheduling. Two notable methods are the Critical Chain Method (CCM), as for example demonstrated by (Long & Ohsato, 2008), and Critical Path (CP), i.e. (Bhaskar et al., 2011). The handbook on project

management and scheduling presents a chapter on fuzzy scheduling including a literature review (Schwindt & Zimmermann, 2015, see page 930-931). The description in this handbook is detailed, stating the most relevant and notable progress in Fuzzy project scheduling up to 2015.

Hence, it is interesting for this thesis to study the literature past the publication date of the handbook as this will include the most recent and novel research. An SLR is conducted on fuzzy project scheduling to identify state of the art fuzzy project scheduling methods that deal with vague information.

F.3 SLR on fuzzy project scheduling

This section presents an SLR conducted on fuzzy project scheduling research between 2015 and July of 2021.

F.3.1 Setup

The setup of the literature study is as follows. The search term query is (*"fuzzy"* **AND** *"project scheduling"*) in the title of the article. The publication date range is set to 01-01-2015 to 01-07-2021 and the language is set to English. The online sources are 'ACM Digital Library', 'IEEE Xplore', 'ProQuest', 'ScienceDirect', 'Scopus', 'SpringerLink', 'Wiley Online Library', and 'WorldCat' The results of the first query are summarized in Table F.1. All unique occurrences are filtered, and the following exclusion rules apply: 1] the topic is fuzzy project scheduling, 2] the publication is available and accessible online and 3] publications should not be part of Schwindt and Zimmermann (2015).

F.3.2 Search results

Table F.1: Search results

Source	**Result**
Scopus	41
WorldCat	15
SpringerLink	7
ProQuest	6
ScienceDirect	6
IEEE Xplore	3
Wiley Online Library	1
ACM Digital Library	0
Total unique results	**44**
Total unique results	44
Excluded by exclusion rule	9
Final result	**35**

The full list of 35 results is presented.

Table F.2: SLR fuzzy scheduling search results

DOI	Citation	Title
10.12988/ams.2015.410852	Elizabeth and Sujatha (2015)	Project scheduling method using triangular intuitionistic fuzzy numbers and triangular fuzzy numbers
10.1007/s00500-014-1395-5	Hu et al. (2015)	An outer–inner fuzzy cellular automata algorithm for dynamic uncertainty multi-project scheduling problem
10.1155/2015/973291	Huang et al. (2015)	An adapted firefly algorithm for product development project scheduling with fuzzy activity duration
10.1016/j.procs.2015.12.359	Knyazeva et al. (2015)	Resource-constrained Project Scheduling Approach under Fuzzy Conditions
#N/A	Krzeszowska-Zakrzewska (2015)	FUZZY PARETO DOMINANCE IN MULTIPLE CRITERIA PROJECT SCHEDULING PROBLEM
10.1111/mice.12166	Ponz-Tienda et al. (2015)	The Fuzzy Project Scheduling Problem with Minimal Generalized Precedence Relations
10.1109/TSMC.2015.2406863	Xu et al. (2015)	A Bilevel Model for Project Scheduling in a Fuzzy Random Environment
#N/A	Azimi and Fathallahi (2016)	Fuzzy multi objective project scheduling under inflationary conditions
10.1080/21681015.2016.1140089	L. Chen and Zhang (2016)	Preemption resource-constrained project scheduling problems with fuzzy random duration and resource availabilities
10.24200/sci.2016.3935	Fathallahi and Najafi (2016)	A hybrid genetic algorithm to maximize net present value of project cash flows in resource-constrained project scheduling problem with fuzzy parameters
10.1007/s40999-016-0031-4	Kaveh et al. (2016)	Fuzzy resource constraint project scheduling problem using CBO and CSS algorithms
10.1016/j.aej.2016.03.038	Tran et al. (2016)	Using Fuzzy Clustering Chaotic-based Differential Evolution to solve multiple resources leveling in the multiple projects scheduling prob lem

Table F.2: SLR fuzzy scheduling results (continued)

DOI	Citation	Title
10.4018/IJFSA.2016100101	Zoulfaghari et al. (2016)	A resource-constrained project scheduling problem with fuzzy activity times
10.1109/CYBConf.2017.7985794	Cheng and Lo (2017)	Multi-project scheduling by fuzzy CA
10.1016/j.procs.2017.11.010	Khalilzadeh et al. (2017)	A Heuristic Algorithm for Project Scheduling with Fuzzy Parameters
10.1007/978-3-030-04491-6_27	Knyazeva et al. (2018)	Modeling decisions for project scheduling optimization problem based on type-2 fuzzy numbers
10.3233/JIFS-171821	Moradi et al. (2018)	Sustainability in fuzzy resource constraint project scheduling in a cooperative environment under uncertainty: Iran's Chitgar lake case study
10.1080/15623599.2016.1225327	Roghanian et al. (2018)	An improved fuzzy critical chain approach in order to face uncertainty in project scheduling
10.1109/ICSSE.2018.8520264	S.-M. Yang et al. (2018)	Integration of Fuzzy Logic and QFD for Critical Chain in Project Scheduling with Uncertainties
10.1007/s40815-018-0542-z	Zhang et al. (2018)	An Effective Solution Approach to Fuzzy Programming with Application to Project Scheduling
10.1007/s40092-019-00328-w	Ahmadpour and Ghezavati (2019)	Modeling and solving multi-skilled resource-constrained project scheduling problem with calendars in fuzzy condition
10.1080/15623599.2019.1619226	Ammar and Abd-ElKhalek (2019)	Criticality measurement in fuzzy project scheduling
10.11121/ijocta.01.2019.00631	Arık (2019)	Credibility based chance constrained programming for project scheduling with fuzzy activity durations
10.1016/j.autcon.2018.11.029	Birjandi and Mousavi (2019)	Fuzzy resource-constrained project scheduling with multiple routes: A heuristic solution
10.6125/JoAAA.201912_51(4).05	Tsai (2019)	Project Scheduling with Resource Constraints by Fuzzy Gantt Chart and Genetic Algorithm
10.1007/s12351-017-0321-x	Alipouri et al. (2020)	A mixed-integer linear programming model for solving fuzzy stochastic resource constrained project scheduling problem
10.1088/1742-6596/1687/1/012011	Dayoub et al. (2020)	Application of Fuzzy theory in project scheduling

Table F.2: SLR fuzzy scheduling results (continued)

DOI	Citation	Title
10.1080/15623599.2020.1828539	Rezakhani (2020)	Hybrid fuzzy-Bayesian decision support tool for dynamic project scheduling and control under uncertainty
10.22111/ijfs.2021.5880	Aramesh et al. (2021)	A new comprehensive project scheduling, monitoring, and management framework based on the critical chain under interval type-2 fuzzy uncertainty
10.1080/01605682.2019.1671150	Liu and Hu (2021)	A dynamic critical path method for project scheduling based on a generalised fuzzy similarity
10.3390/su13052801	Kuchta et al. (2021)	Sustainability-oriented project scheduling based on z-fuzzy numbers for public institutions
10.1016/j.asoc.2020.106988	Mahdavi et al. (2021)	Toward a scalable type-2 fuzzy model for resource-constrained project scheduling problem
10.1108/ECAM-07-2020-0540	Rezakhani (2021)	Project scheduling and performance prediction: a fuzzy-Bayesian network approach
10.1145/3448748.3448793	H. Yang et al. (2021)	Fuzzy Resource Constrained Project Scheduling Optimization with Hybrid Multiobjective Genetic Algorithm
10.1016/j.cie.2021.107316	Yuan et al. (2021)	Multi-objective multi-mode resource-constrained project scheduling with fuzzy activity durations in prefabricated building construction

F.3.3 Result characteristics

A list of characteristics per publication is listed.

Table F.3: Characteristics of publications on fuzzy project scheduling

Citation	Field of application	PSP type	Fuzzy applied to	Fuzzy type	Solution method
Elizabeth and Sujatha (2015)	Unspecified	Fuzzy critical path method	Edge weights or arc lengths	Triangular intuitionistic fuzzy numbers and triangular fuzzy numbers	Ranking
Hu et al. (2015)	Lean management	Dynamic uncertain multi-project scheduling problem	Project cell and process cell	Fuzzy cell	Outer–inner uncertainty fuzzy cellular automata algorithm
Huang et al. (2015)	Product development	Fuzzy RCPSP	Activity duration	Trapezoidal fuzzy number	Adapted fuzzy firefly algorithm, fuzzy parallel SGS
Knyazeva et al. (2015)	Unspecified	Fuzzy RCPSP	Fuzzy activity durations, fuzzy activity network with an adjacency matrix	Flat fuzzy number with five linear pieces	Ranking and heuristic priority rule
Krzeszowska-Zakrzewska (2015)	MCDMP	Multi criteria PSP	Pareto dominance relation		evolutionary algorithm SPEA2 and fuzzification of Pareto dominance
Ponz-Tienda et al. (2015)	Construction	Fuzzy PSP with generalized precedence and flow relationships	precedence relationships	values	heuristic algorithm

Table F.3: Characteristics of papers on fuzzy project scheduling (continued)

Citation	Field of application	PSP type	Fuzzy applied to	Fuzzy type	Solution method
Xu et al. (2015)	Unspecified	Multiple decision-maker PSP	random objective function and constraints	Fuzzy random membership function	Fuzzy random simulation-based bilevel global-local-neighbor PSO
Azimi and Fathallahi (2016)	Unspecified	PSP with discounted cash flows	Activity duration and sets under inflationary condition	Triangular fuzzy numbers	LP-Metric method and Frank-Wolf
L. Chen and Zhang (2016)	Unspecified	Preemption RCPSP	Random duration and resource availabilities		local search PSO
Fathallahi and Najafi (2016)	Unspecified	RCPSP under uncertainty with discounted cash flows	duration, cash flows, resource consumption, and interest rate	fuzzy numbers	tuned hybrid GA with SA
Kaveh et al. (2016)	Unspecified	Fuzzy RCPSP	Activity duration	Fuzzy numbers	CSS CBO
Tran et al. (2016)	Construction	PSP with leveling of multiple resources in multi projects	#N/A	fuzzy c-means clustering	Fuzzy Clustering Chaotic-based DE
Zoulfaghari et al. (2016)	Unspecified	RCPSP	Activity times	Triangular fuzzy numbers	Multi-objective linear programming model
Cheng and Lo (2017)	Unspecified	Multi-mode RCMPSP	Soft capacity constraints	Fuzzy CA	Hierarchical decision-making, CA
Khalilzadeh et al. (2017)	Unspecified	RCPSP	Activity durations and resource limitations	Fuzzy numbers	parallel SGS

Table F.3: Characteristics of papers on fuzzy project scheduling (continued)

Citation	Field of application	PSP type	Fuzzy applied to	Fuzzy type	Solution method
Knyazeva et al. (2018)	Agricultural production	MCDMP	Uncertain variables	Type-2 fuzzy numbers	Heuristic decision algorithm with enumeration tree and partial schedules
Moradi et al. (2018)	Unspecified	RCPSP under uncertainty	modes of activities	linear membership function	cooperative game methods
Roghanian et al. (2018)	Unspecified	RCPSP		Fuzzy number	CCM with buffer sizing using square root of sum of squares method
S.-M. Yang et al. (2018)	Unspecified	CCM	Project time limit, activity begin time, budget, manpower, technical difficulty, and facility requirement	trapezoidal, triangular, or Gaussian fuzzy numbers	CCM
Ahmadpour and Ghezavati (2019)	Unspecified	RCPSP with staff skill levels	Manpower,	Triangular membership function	Exact
Arık (2019)	Unspecified	RCPSP	activity duration	Triangular membership function	Credibility based chance constrained programming

Table F.3: Characteristics of papers on fuzzy project scheduling (continued)

Citation	Field of application	PSP type	Fuzzy applied to	Fuzzy type	Solution method
Birjandi and Mousavi (2019)	Unspecified	RCPSP with flexible activities and multiple routes	activity duration, non-renewable resource	trapezoidal fuzzy numbers	fuzzy mixed integer nonlinear programming using hybrid meta-heuristic, being a distribution rule heuristic, binary PSO and GA
Tsai (2019)	Unspecified	RCPSP	Gantt chart for resource requirement	Triangular membership function	GA to determine priority and duration
Alipouri et al. (2020)	Unspecified	RCPSP with mixed uncertainty (randomness and fuzziness)	activity duration	Trapezoidal and triangular membership functions	MILP
Rezakhani (2020)	Unspecified	Dynamic project scheduling and control under uncertainty	Expert evaluations	Type-2 interval-valued fuzzy numbers	Bayesian network
Aramesh et al. (2021)	Unspecified	RCPSP with time/resource trade-offs	Interval uncertainty	Type-2 fuzzy numbers	CM method using SA
Mahdavi et al. (2021)	Unspecified	RCPSP	activity time span	type-2 fuzzy numbers	Dijkstra algorithm, type-2 fuzzy simplex algorithm
H. Yang et al. (2021)	Unspecified	Fuzzy RCPSP	activities with random duration	Triangular membership function	hybrid multi-objective GA NSGA-II

Table F.3: Characteristics of papers on fuzzy project scheduling (continued)

Citation	Field of application	PSP type	Fuzzy applied to	Fuzzy type	Solution method
Yuan et al. (2021)	Construction	multi-objective multi-mode	activity durations RCPSP	triangular fuzzy number	hybrid cooperative co-evolution algorithm

The conclusion is that virtually all methods require multiple inputs (such as triangular fuzzy numbers). Getting that additional information from employees at JIVC is currently infeasible. Therefore, fuzzy project scheduling is seen as an unsuitable solution.

Appendix G

Summaries

This appendix presents a summary for each of the eight publications in subsection G.0.1. First an overview is presented in Table G.1.

G.0.1 Characteristics

All eight publications are summarized (Appendix G) and the characteristics are presented next (Table G.1).

Table G.1: Characteristics of publications on combinatorial auctions and negotiations in project scheduling

Citation	Field of application	PSP type	Negotiation or auction type	Agent types	Solution method	Dataset
Lau et al. (2005)	Supply chain in MTO production	Distributed PSP	Negotiation with schedule flexibility information	Project; middle; contractor;	left and right-shift heuristics, insertion heuristic, voting	Custom
Li (2009)	Shipbuilding	Ship multi-project scheduling	Negotiation using MCNM	Project-coordination; ship project; process; feed-preparation; critical resources; monitoring; peripheral cooperation	SA, GA, NN, based on FIS	#N/A
Adhau et al. (2012)	Unspecified	DRCMPSP	Negotiation and iterative ascending CA	Project; resource; exchange	Parallel SGS	MPSPLIB
Song et al. (2016) and Song et al. (2017)	Unspecified	DRCMPSP	Multi-unit CA	Project; mediator	Capacity query through greedy heuristic and branch-and-bound improvement algorithm, parallel SGS	MPSPLIB
Cheng and Lo (2017) and Cheng et al. (2019)	Unspecified	Multi-mode and fuzzy RCMPSP	CA, fuzzy CA	Project; manager	Bi-level decentralized programming problem	Modified PSPLIB, MMLIB
Homberger and Fink (2017)	Unspecified	DRCMPSP	Generic negotiation mechanisms with side payments	Regular; mediator	ACO	Modified PSPLIB

Lau et al. (2005) is the first part of a series of two papers in which they present an agent based negotiation model. In it, project agents negotiate with contract agents via intermediate middle agents. They utilize 'schedule flexibility information', which is partial information shared during negotiation between agents, being: start of a time window, length of the time window and costs for violating the window. Agents have personal objectives which they try to optimize. For project agents, this is the sum of project operating cost and costs of violating time windows proposed by contract agents. For contract agents, this is the sum of revenue of project agent operations minus the costs of violating time windows proposed by project agents. Prices are calculated by left- and right shifting and schedules are chosen by means of voting. The method in this paper may be less suited due to its main application in supply chains. Nonetheless, the concept of schedule flexibility information is useful because it reveals that thinking in terms of starting windows is superior to that of one fixed starting epochs.

Li (2009) present a negotiation model with a three level hierarchy specifically for project scheduling in ship building. Negotiation is used to allocate resource, specifically using Modified Contract Net Mechanism (MCNM). The components in this paper are only discussed in broad terms and almost presented as a framework. As many as seven agent types are introduced in this method. The method presented in this paper may be less suited due to its main application in ship building, nonetheless, it shows that there is no limit to the number of agent types one can introduce to further distribute decision making and complex problem solving.

Adhau et al. (2012) present an iterative auction based negotiation having project agents, resource agents and exchange agents that oversee the negotiation. Project agents bid for resources, resource agents calculate resource prices and exchange agents coordinate the auction. Utility costs are calculated by considering the cost of other resources that are waiting (idleness), the tardiness cost of the project activity, and the cost of the resource. Next, project agents bid prices based on scheduling their own project using a parallel SGS. Provisional winners are determined by filtering the bids on feasibility and selecting the best price of remaining bids. Next, utilization and prices are adjusted and another round is held until price stagnation. This interesting paper introduces a combination of a CA and negotiation.

The publications of Song et al. (2016) and Song et al. (2017) both regard the decentralized RCMPSP and focus on computational efficiency to solve large problem instances in reasonable amount of computational time. Both publications employ a custom multi-unit CA framework based on that of Confessore et al. (2006), but use different methods to organise the auction. In the first publication, a greedy resource allocation strategy is used, where bids and demand ratio's are communicated between agents and an additional bid modification step is used to improve resource utilization. The latter publication, a capacity query method is used to compute valuations, followed by a greedy allocation method which is in turn further improved by a branch-and-bound bid sequencing algorithm. Furthermore, the latter article especially elaborates on the mathematical proof of the auction mechanism. These publications are particularly interesting because the clear description of the computational difficulty of problems that arise in CA's and ability to efficiently deal with large problem instances.

The publications of Cheng and Lo (2017) and Cheng et al. (2019) both regard the multi-mode decentralized RCMPSP and employ a CA and fuzzy CA. The decision making

is formulated as a bi-level decentralized programming problem. Project agents solve a local scheduling problem and generate multiple possible schedules. The manager agent attempts to find a feasible combination of bids. If no feasible solution is found, due dates are extended gradually. If this still does not result in a feasible solution, a fuzzy CA is organized which proposes how to expand resources such that the plan will become feasible. The multi-mode aspect seems less interesting to the case in this thesis, nonetheless, the method can find solutions that (slightly) violate feasibility constraints, which may be useful in real-life scenarios.

Homberger and Fink (2017) present a negotiation based method and use monetary values in their negotiation. Using such a single unit in their offers enables to use of cash flows and side payments. A random solution to the problem is generated to initiate the negotiation process. Afterwards, new random solutions are generated and agents negotiate whether or not this solution is better, thus the procedure maximizes social welfare.

Appendix H

Problem instance

An example is given of a full problem instance, using that of **mp_j30_a2_nr2** as an example throughout the thesis. An MPSPLib instance consists of a multi-project instance file (section H.1), multiple single project instances files (section H.2) and additional information not included in those files (section H.3). A mathematical description is given of an input instance (section H.4).

H.1 MPSPLib Multi-Project instance

Each MPSPLib instance is represented as an .xml file (Listing H.1).

```
<!DOCTYPE mp-list SYSTEM "mp.dtd">
<mp-list>
    <mp>
        <name>mp_j30_a2_nr2</name>
        <project-list>
            <project>
                <filename>KolischInstanzen/j30/j309_9.sm</filename>
                <start>0</start>
            </project>
            <project>
                <filename>KolischInstanzen/j30/j3033_3.sm</filename>
                <start>5</start>
            </project>
        </project-list>
        <resources>
            <resource>21</resource>
            <resource>0</resource>
            <resource>0</resource>
            <resource>0</resource>
        </resources>
    </mp>
</mp-list>
```

Listing H.1: MPSPLib problem instance

H.2 Kolisch RCPSP instances

Each single project instance is represented as a .sm file which is a custom extension. It is encoded in ASCII and can therefore easily be read as a .txt-file. The two instances used by the multi-project instance are presented here. Not all information in these instances is relevant or used by the method. Moreover, the release date is overwritten and the resource availabilities are overwritten for all resources that are considered global.

H.2.1 j309_9

```
******************************************************************************
file with basedata            : j30_25.bas
initial value random generator: 858231297
******************************************************************************
projects                      :  1
jobs (incl. supersource/sink ):  32
horizon                       :  135
RESOURCES
  - renewable                 :  4   R
  - nonrenewable              :  0   N
  - doubly constrained        :  0   D
******************************************************************************
PROJECT INFORMATION:
pronr.  #jobs rel.date duedate tardcost  MPM-Time
    1     30      0       37       20       37
******************************************************************************
PRECEDENCE RELATIONS:
jobnr.    #modes  #successors   successors
   1        1          3           2   3   4
   2        1          2           6   9
   3        1          1          11
   4        1          3           5   7  17
   5        1          3           8  13  22
   6        1          1          12
   7        1          3          10  19  28
   8        1          1          29
   9        1          3          13  18  20
  10        1          1          18
  11        1          2          16  31
  12        1          1          15
  13        1          2          14  19
  14        1          1          26
  15        1          1          24
  16        1          1          28
  17        1          1          21
  18        1          1          23
  19        1          1          27
  20        1          2          22  25
  21        1          2          26  27
  22        1          1          26
  23        1          2          27  30
  24        1          2          25  30
  25        1          1          31
  26        1          1          28
  27        1          1          29
  28        1          1          29
  29        1          1          32
  30        1          1          32
  31        1          1          32
  32        1          0
******************************************************************************
```

Listing H.2: j309_9 problem instance

```
REQUESTS/DURATIONS:
jobnr. mode duration  R 1  R 2  R 3  R 4
------------------------------------------------------------------------
  1      1     0       0    0    0    0
  2      1     5       5    0    8    3
  3      1     4       6    7    8    0
  4      1     7       8    0    4    5
  5      1     3       3    9    2    8
  6      1    10       5    3    6    9
  7      1     4       1    8    1    8
  8      1     2       5    5    0    3
  9      1     7       2    3    0    0
 10      1     7       1    0    0    8
 11      1     3       5    9    0    6
 12      1     2       2   10    4    0
 13      1     4       0    4    7    4
 14      1     2       3    8    2    0
 15      1     2       0    1    0    2
 16      1     7       3    9    5    6
 17      1     5       9    0    0   10
 18      1     3       6   10    9    0
 19      1     6       5    6    0    8
 20      1     4       5    3    9    0
 21      1     6       8    8   10    8
 22      1     5       1    4    0    2
 23      1     7       0    0   10    2
 24      1     3       5    8    0    6
 25      1     2       0   10    8    4
 26      1     6       4    9   10    2
 27      1     1       1    7    3    2
 28      1     4       0    4    6    5
 29      1     4       5    0    8    0
 30      1     9       0    6    4    3
 31      1     1       0    5    1    6
 32      1     0       0    0    0    0
************************************************************************
RESOURCEAVAILABILITIES:
  R 1  R 2  R 3  R 4
   12   15   15   16
************************************************************************
```

Listing H.3: j309_9 problem instance (continued)

H.2.2 j3033_3

```
******************************************************************************
file with basedata            : j30_49.bas
initial value random generator: 1515273925
******************************************************************************
projects                      :  1
jobs (incl. supersource/sink ):  32
horizon                       :  169
RESOURCES
  - renewable                 :  4   R
  - nonrenewable              :  0   N
  - doubly constrained        :  0   D
******************************************************************************
PROJECT INFORMATION:
pronr.  #jobs rel.date duedate tardcost  MPM-Time
    1     30      0       42        9       42
******************************************************************************
PRECEDENCE RELATIONS:
jobnr.    #modes  #successors   successors
   1        1          3           2   3   4
   2        1          3           5  11  17
   3        1          3           7  10  13
   4        1          3           8  11  17
   5        1          3           6   9  20
   6        1          3           8  12  21
   7        1          3           8  15  20
   8        1          3          14  19  24
   9        1          3          15  16  21
  10        1          3          11  17  20
  11        1          2          12  16
  12        1          2          18  25
  13        1          3          19  24  27
  14        1          3          22  25  27
  15        1          2          23  26
  16        1          3          18  25  27
  17        1          2          23  30
  18        1          3          19  22  24
  19        1          1          31
  20        1          3          23  26  29
  21        1          2          22  29
  22        1          1          30
  23        1          1          31
  24        1          1          26
  25        1          1          28
  26        1          2          28  30
  27        1          2          28  29
  28        1          1          31
  29        1          1          32
  30        1          1          32
  31        1          1          32
  32        1          0
******************************************************************************
```

Listing H.4: j3033_3 problem instance

```
REQUESTS/DURATIONS:
jobnr. mode duration  R 1  R 2  R 3  R 4
------------------------------------------------------------------------
  1      1     0       0    0    0    0
  2      1     1       0    0    0    1
  3      1     2       0    2    0    0
  4      1     9       0    9    0    0
  5      1     7       0    9    0    0
  6      1     1       9    0    0    0
  7      1     9       0    0    8    0
  8      1     7       0    0    0    3
  9      1     5       9    0    0    0
 10      1     4       0    0    6    0
 11      1     1       4    0    0    0
 12      1     7       0    0    7    0
 13      1     3       0    0    0    5
 14      1     3       0    0    2    0
 15      1    10       0    0    1    0
 16      1     2       0    2    0    0
 17      1    10       1    0    0    0
 18      1     6       0    0    7    0
 19      1     8       0    7    0    0
 20      1    10       0    0    0    6
 21      1     2       0    0    4    0
 22      1     2       0    0    1    0
 23      1     6       0    0    0    1
 24      1     7       9    0    0    0
 25      1    10       4    0    0    0
 26      1     2       0    0    0    3
 27      1     7       7    0    0    0
 28      1     4       0    0    1    0
 29      1    10       0    1    0    0
 30      1    10       1    0    0    0
 31      1     4       0    0    3    0
 32      1     0       0    0    0    0
************************************************************************
RESOURCEAVAILABILITIES:
  R 1  R 2  R 3  R 4
   11   11    9    7
************************************************************************
```

Listing H.5: j3033_3 problem instance (continued)

H.3 Additional information

The information as provided in both files are not sufficient and does not complete a problem instance as used by Song et al. (2016). Additional parameters are the revenue at completion per project and the unit delay penalty (or tardiness costs) per project. It is unspecified in the publication if the unit delay penalty is based on the "tardcost" parameter or if it is provided by another source. Moreover, the revenue at completion is not provided by the publication.

H.4 Mathematical description input

A mathematical description is given of the input, consisting of sets (subsection H.4.1), decision variables (subsection H.4.2), and input variables (subsection H.4.3).

H.4.1 Sets

Seven sets are stated (Table H.1). The number of tasks and local instances could deviate for each project, although this is not the case for the MPSPLib instances. The variable T is the time horizon.

Table H.1: Sets

Symbol		Set description	Set
$\mathcal{J}$		Projects	$j \in \{1, ..., \lvert\mathcal{J}\rvert\}$
$\mathcal{I}_j$	$\forall j \in \mathcal{J}$	Tasks	$i \in \{1, ..., \lvert\mathcal{I}_j\rvert\}$
$\mathcal{K}g$		Global resources	$k \in \{1, ..., \lvert\mathcal{K}g\rvert\}$
$\mathcal{K}l_j$	$\forall j \in \mathcal{J}$	Local resources	$k_l \in \{1, ..., \lvert\mathcal{K}l_j\rvert\}$
$Pred_{i,j}$	$\forall i \in \mathcal{I}_j, j \in \mathcal{J}$	Predecessors	$Pred_{i,j} \subset \mathcal{I}_j \setminus \{i\}$
$Succ_{i,j}$	$\forall i \in \mathcal{I}_j, j \in \mathcal{J}$	Successors	$Succ_{i,j} \subset \mathcal{I}_j \setminus \{i\}$
$\mathcal{T}$		Time	$t \in 0, ..., T$

H.4.2 Decision variables

The only decision variables are the starting times of the tasks (Table H.2). These result in a schedule if aggregated per project, or in a solution if aggregated for all projects in the instance.

Table H.2: Decision variables

Symbol		Description	Range
$S_{i,j}$	$\forall i \in \mathcal{I}_j, j \in \mathcal{J}$	Start time	$\{S_{i,j} \in \mathbb{N}_0 \lvert esd_j \leq S_{i,j} < T\}$
S_j	$\forall j \in \mathcal{J}$	Schedule	$(S_{1,j}, ..., S_{\lvert\mathcal{I}_j\rvert,j})$
S		Solution	$(S_1, ..., S_{\lvert\mathcal{J}\rvert})$

H.4.3 Input variables

The primary problem instance input variables are presented (Table H.3).

Table H.3: Input variables

Symbol		**Description**	**Range**
T		Time horizon	$T \in \mathbb{N}_0$
Rg_k	$\forall k \in \mathcal{K}g$	Global resource level	$Rg_k \in \mathbb{N}_0$
Rl_{j,k_l}	$\forall k_l \in \mathcal{K}l_j, j \in \mathcal{J}$	Local resource level	$Rl_{j,k_l} \in \mathbb{N}_0$
$d_{i,j}$	$\forall i \in \mathcal{I}_j, j \in \mathcal{J}$	Task duration	$d_{i,j} \in \mathbb{N}_0$
$rg_{i,j,k}$	$\forall i \in \mathcal{I}_j, j \in \mathcal{J}, k \in \mathcal{K}g,$	Global resource requirement	$\{rg_{i,j,k} \in \mathbb{N}_0 \vert rg_{i,j,k} \leq Rg_k\}$
rl_{i,j,k_l}	$\forall i \in \mathcal{I}_j, j \in \mathcal{J}, k_l \in \mathcal{K}l_j$	Local resource requirement	$\{k_l \in \mathbb{N}_0 \vert rl_{i,j,k_l} \leq Rl_{j,k_l}\}$
esd_j	$\forall j \in \mathcal{J}$	Earliest start date	$\{esd_j \in \mathbb{N}_0 \vert esd_j < T\}$
rev_j	$\forall j \in \mathcal{J}$	Revenue at completion	$rev_j \in \mathbb{Q}^+$
udp_j	$\forall j \in \mathcal{J}$	Unit delay penalty	$udp_j \in \mathbb{Q}^+$

Appendix I

Characteristics

The characteristics are split in a public and private part (section I.1 and section I.2 respectively).

I.1 Public

The publicly available characteristics are saved in a .txt file (Listing I.1).

```
Overload factor               : 71.02%
Average utilization factor GR0: 71.02%
```

Listing I.1: Global characteristics

I.2 Private

A part of the output is private information that is only accessible for the separate project agents. This instance contains two projects, being j309_9 (subsection I.2.1) and j3033_3 (subsection I.2.2).

I.2.1 j309_9

The private information consists of a .txt file containg information (Listing I.2) and the critical schedule (Figure I.1).

```
Critical path duration        : 37 days
Average utilization factor LR0: 111.35%
Average utilization factor LR1: 109.91%
Average utilization factor LR2: 101.52%
```

Listing I.2: Characteristics of j309_9

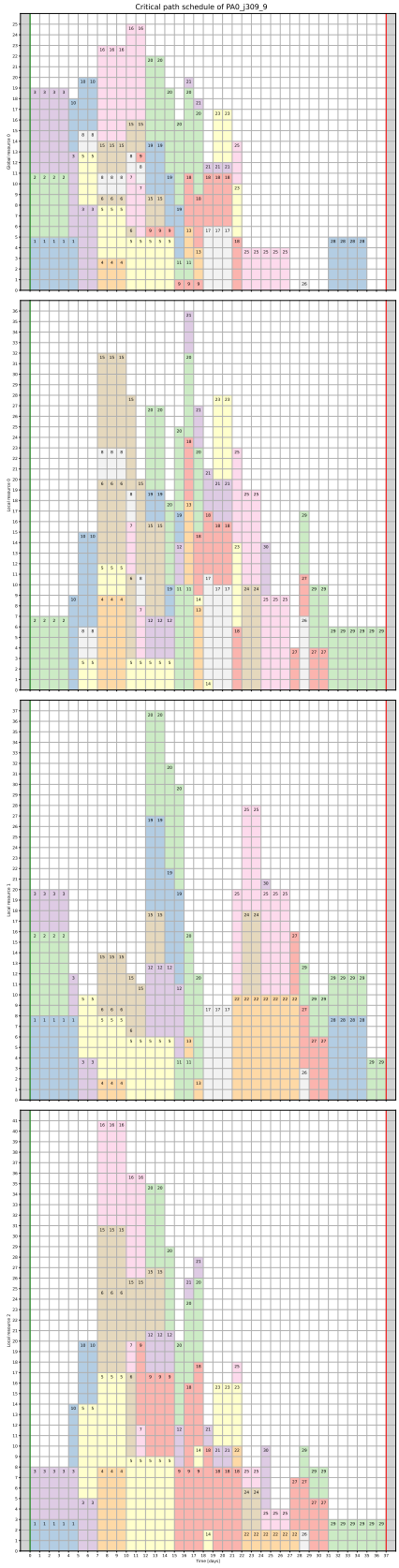


Figure I.1: Critical schedule of project instance j309_9

I.2.2 j3033_3

The private information consists of a .txt file containg information (Listing I.3) and the critical schedule (Figure I.2).

```
Critical path duration        : 42 days
Average utilization factor LR0: 47.19%
Average utilization factor LR1: 60.58%
Average utilization factor LR2: 37.07%
```

Listing I.3: Characteristics of j3033_3

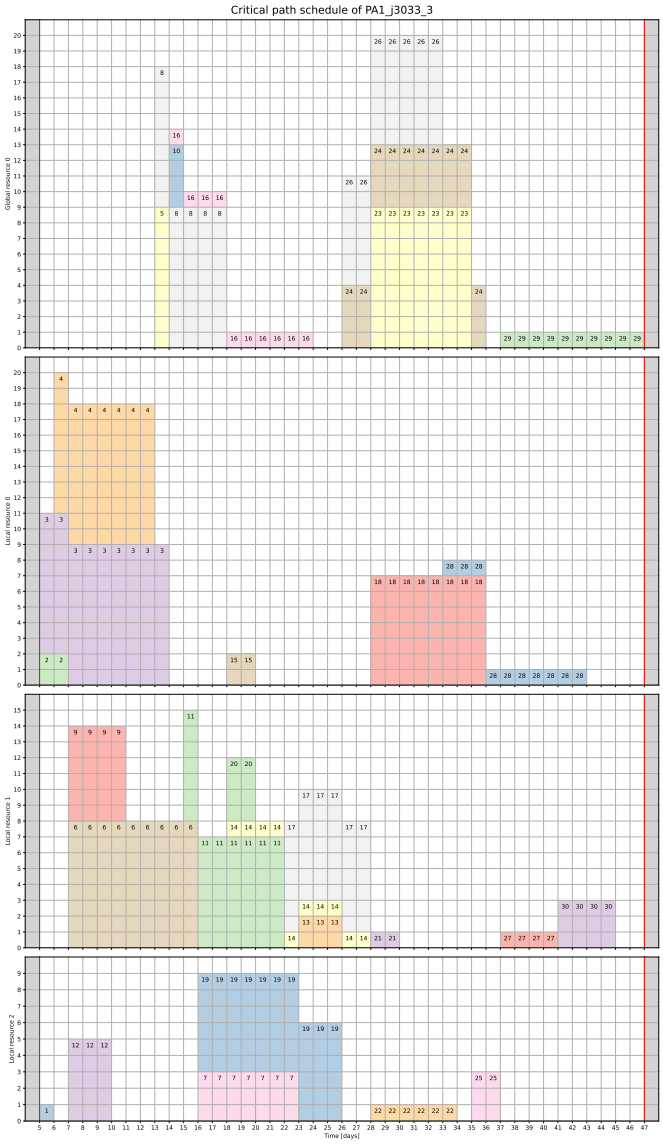


Figure I.2: Critical schedule of project instance j3033_3

Appendix J

Output

The output is split in a public and private part (section J.1 section J.2 respectively). Additionaly, a log file is generated.

J.1 Public

A part of the output is public information. It consists of a .txt file of performance measures (Listing J.1), a .pdf plot of global resource usage (Figure J.1), and a .pdf plot of resource utilization (Figure J.2).

```
Average makespan     : 72.5 days
Total makespan       : 84 days
Average project delay: 33.0 days
Std of project delay : 19.80 days
Computing time       : 1.457 seconds
Resource utilization : (True, 0) 39.74%
```

Listing J.1: Global performance measures

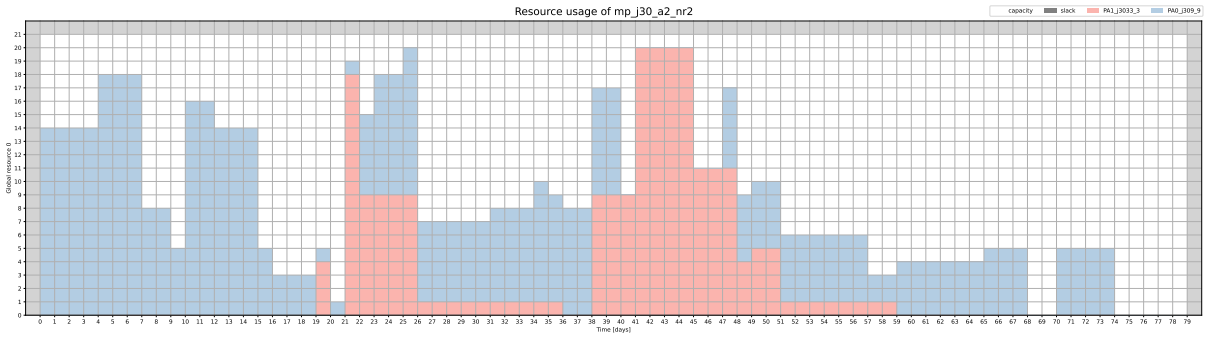


Figure J.1: Global resource usage

J.2 Private

A part of the output is private information that is only accessible for the separate project agents. This instance contains two projects, being j309_9 (subsection J.2.1) and j3033_3

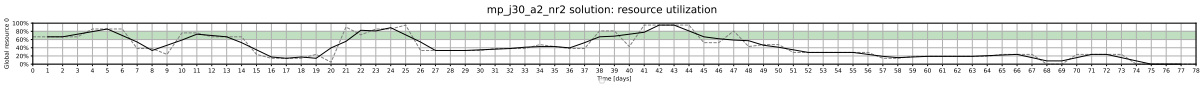


Figure J.2: Global resource utilization

(subsection J.2.2).

J.2.1 j309_9

The output consists of a .txt file of performance measures (Listing J.2), a .pdf plot of global resource usage (Figure J.3), and a .pdf plot of resource utilization (Figure J.4).

```
Makespan                   : 79 days
Start delay                : 0 days
Project delay              : 42 days
Combined slack sufficiency : 367
Relative gap               : 113.51%
```

Listing J.2: Performance measures of j309_9

J.2.2 j3033_3

The output consists of a .txt file of performance measures (Listing J.3), a .pdf plot of global resource usage (Figure J.5), and a .pdf plot of resource utilization (Figure J.6).

```
Makespan                   : 54 days
Start delay                : 0 days
Project delay              : 12 days
Combined slack sufficiency : 228
Relative gap               : 28.57%
```

Listing J.3: Performance measures of j3033_3

J.3 Log

A log is presented indicating what each agent is doing (Listing J.4). It can be used for debugging purposes and for analytic purposes, for example to see agent decision making.

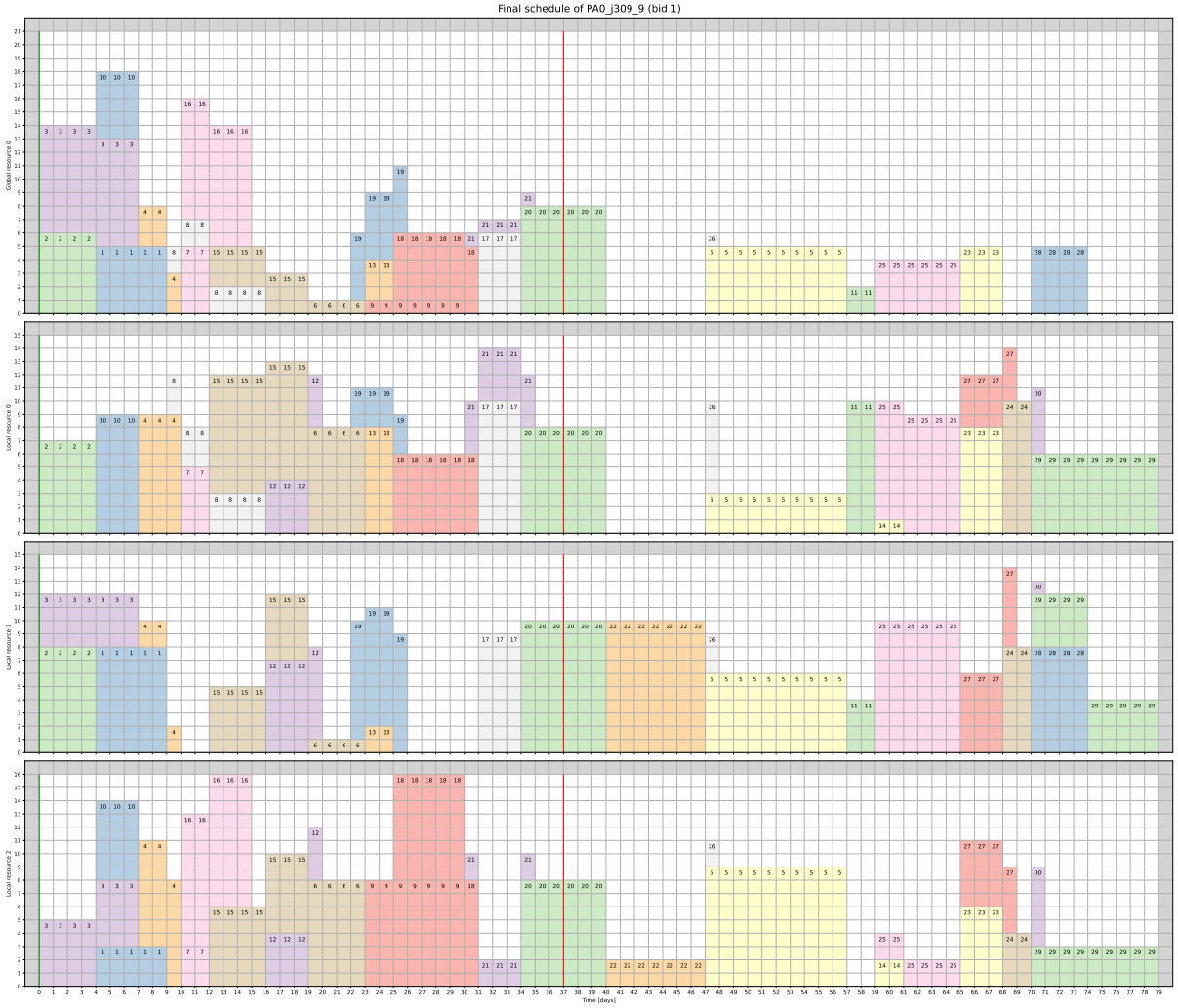


Figure J.3: Final schedule of project instance j309_9

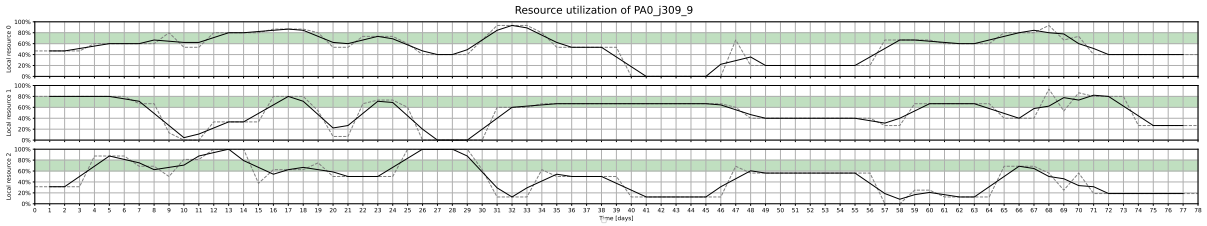


Figure J.4: Resource usage of project instance j309_9

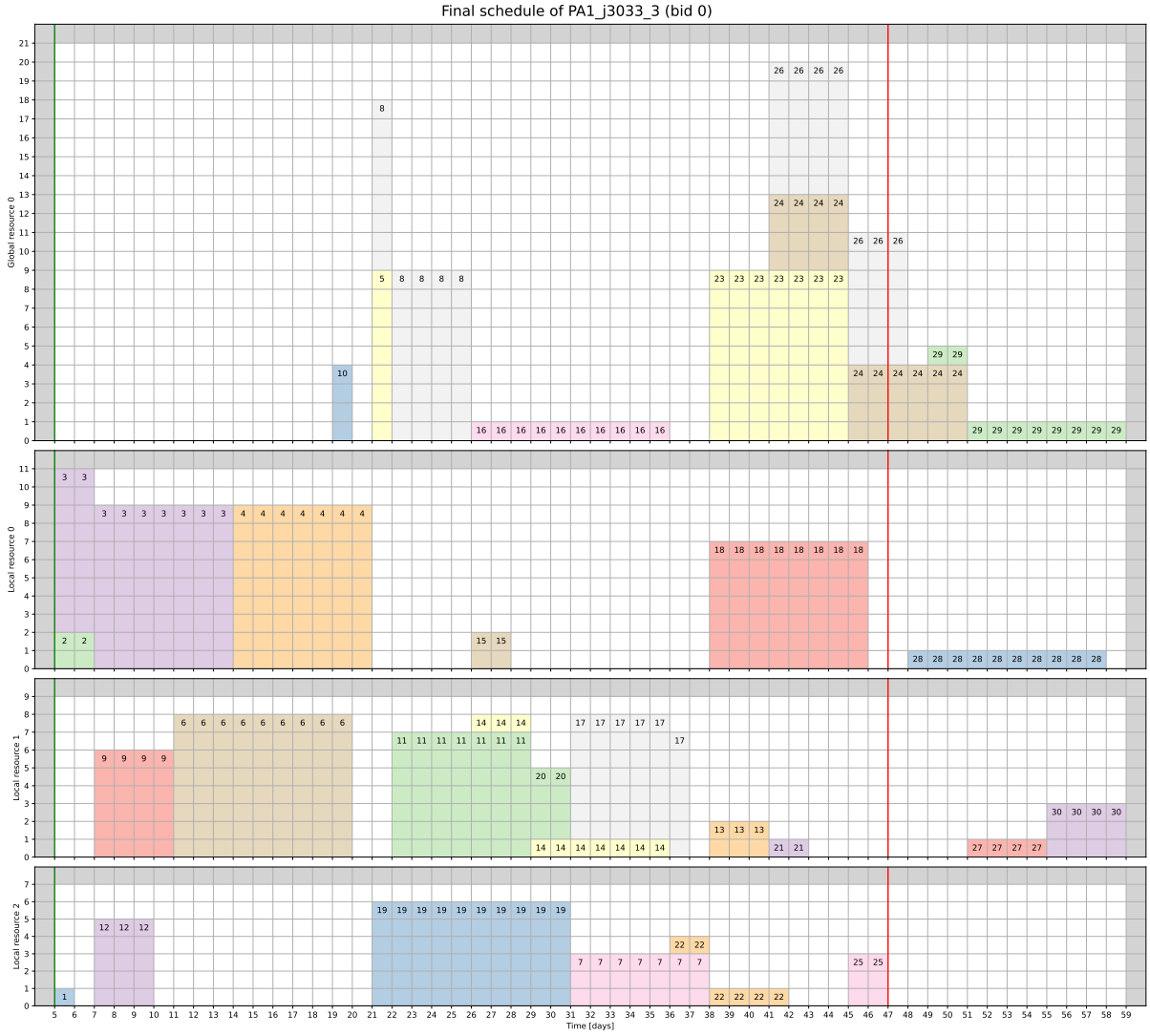


Figure J.5: Final schedule of project instance j3033_3

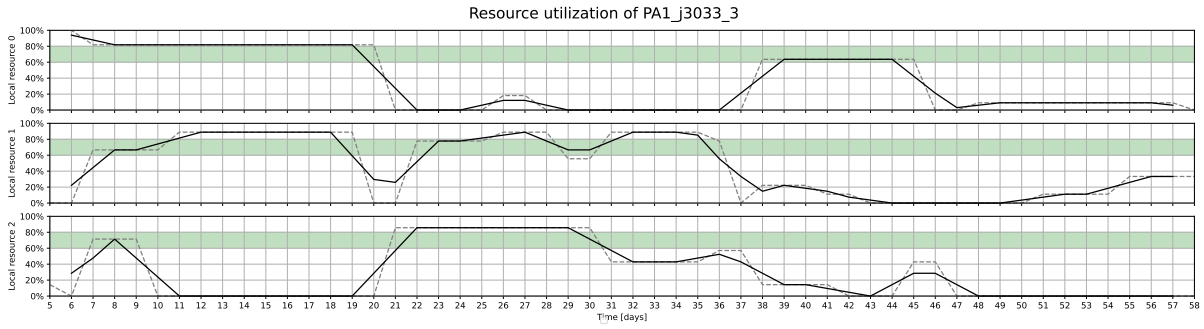


Figure J.6: Resource usage of project instance j3033_3

```
Auction:  ==== START ====
Auction.Auctioneer: Initializing
Auction.Auctioneer: Importing auction
Auction.Auctioneer: Importing auction: resource agent
Auction.ResourceAgent: Initializing
Auction.Auctioneer: Importing auction: project agents
Auction.Auctioneer: Importing auction: project agent PA0_j309_9
Auction.PA0_j309_9: Initializing
Auction.PA0_j309_9: Generating critical schedule
Auction.PA0_j309_9: Plotting critical schedule
Auction.PA0_j309_9: Presenting characteristics
Auction.Auctioneer: Importing auction: project agent PA1_j3033_3
Auction.PA1_j3033_3: Initializing
Auction.PA1_j3033_3: Generating critical schedule
Auction.PA1_j3033_3: Plotting critical schedule
Auction.PA1_j3033_3: Presenting characteristics
Auction.Auctioneer: Importing auction completed
Auction.Auctioneer: Starting auction with 2 bidder(s)
Auction.Auctioneer: 2 bidder(s) bidding in initial phase 0
Auction.PA1_j3033_3: Generating bid 0 by solving bidding problem
Auction.PA1_j3033_3: Schedule has combined slack sufficiency of 214
Auction.Auctioneer: Received bid from PA1_j3033_3 with valuation 9983.0
Auction.PA0_j309_9: Generating bid 0 by solving bidding problem
Auction.PA0_j309_9: Schedule has combined slack sufficiency of 342
Auction.Auctioneer: Received bid from PA0_j309_9 with valuation 9964.0
Auction.Auctioneer: Determining winning bidder(s) by solving WDP
Auction.Auctioneer: Accepting initial bid of PA1_j3033_3 with criteria value 3016.52
Auction.Auctioneer: Rejecting initial bid of PA0_j309_9 with criteria value 2103.94
Auction.Auctioneer: Calculating demand ratio 0
Auction.Auctioneer: 1 bidder(s) bidding in final phase 0
Auction.PA1_j3033_3: Generating modified bid 0 by solving bid modification problem
Auction.Auctioneer: Determining winning bidder(s) by solving WDP
Auction.Auctioneer: Identifying winner: first subtracting resources
Auction.ResourceAgent: Subtracting allocated resources from available resources
Auction.Auctioneer: Updating available global resource level to 1
Auction.Auctioneer: Accepting final bid of PA1_j3033_3 with criteria value 2947.03
Auction.PA1_j3033_3: Presenting output
Auction.Auctioneer: 1 bidder(s) bidding in initial phase 1
Auction.PA0_j309_9: Generating bid 0 by solving bidding problem
Auction.PA0_j309_9: Schedule has combined slack sufficiency of 377
Auction.Auctioneer: Received bid from PA0_j309_9 with valuation 9958.0
Auction.Auctioneer: Determining winning bidder(s) by solving WDP
Auction.Auctioneer: Accepting initial bid of PA0_j309_9 with criteria value 1979.41
Auction.Auctioneer: Calculating demand ratio 1
Auction.Auctioneer: 1 bidder(s) bidding in final phase 1
Auction.PA0_j309_9: Generating modified bid 1 by solving bid modification problem
Auction.Auctioneer: Determining winning bidder(s) by solving WDP
Auction.Auctioneer: Identifying winner: first subtracting resources
Auction.ResourceAgent: Subtracting allocated resources from available resources
Auction.Auctioneer: Updating available global resource level to 2
Auction.Auctioneer: Accepting final bid of PA0_j309_9 with criteria value 1885.79
Auction.PA0_j309_9: Presenting output
Auction.Auctioneer: Closing auction and presenting output
Auction:  ===== END =====
```

Listing J.4: Log of auction

Appendix K

Output robust

The output is split in a public and private part (section K.1 and section K.2 respectively). Additionally, a log file is generated (section K.3).

K.1 Public

A part of the output is public information. It consists of a .txt file of performance measures (Listing K.1), a .pdf plot of global resource usage (Figure K.1), and a .pdf plot of resource utilization (Figure K.2).

```
Average makespan     : 72.5 days
Total makespan       : 84 days
Average project delay: 33.0 days
Std of project delay : 19.80 days
Computing time       : 1.457 seconds
Resource utilization : (True, 0) 39.74%
```

Listing K.1: Global performance measures

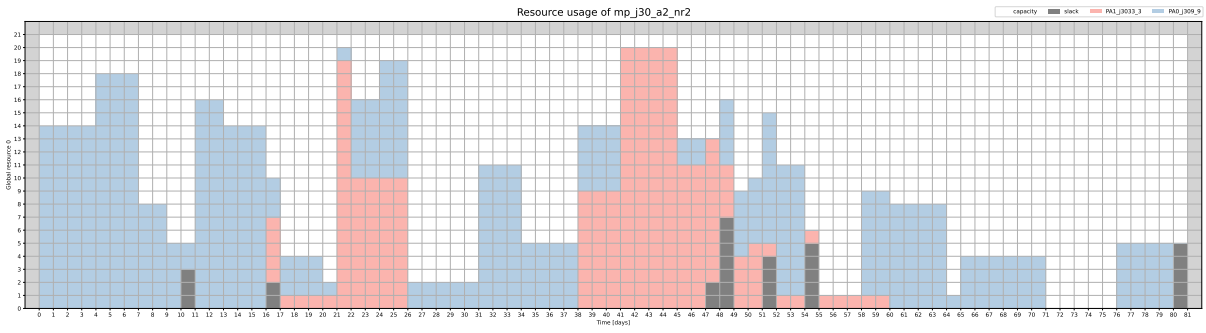


Figure K.1: Global resource usage

K.2 Private

A part of the output is private information that is only accessible for the separate project agents. This instance contains two projects, being j309_9 (subsection K.2.1) and j3033_3

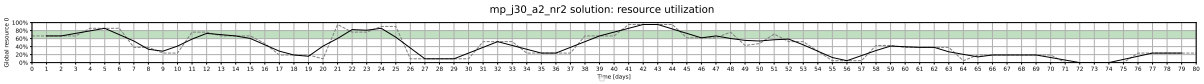


Figure K.2: Global resource utilization

(subsection K.2.2).

K.2.1 j309_9

The output consists of a .txt file of performance measures (Listing K.2), a .pdf plot of global resource usage (Figure K.3), and a .pdf plot of resource utilization (Figure K.4).

```
Makespan                    : 79 days
Start delay                 : 0 days
Project delay               : 42 days
Combined slack sufficiency : 367
Relative gap                : 113.51%
```

Listing K.2: Performance measures of j309_9

K.2.2 j3033_3

The output consists of a .txt file of performance measures (Listing K.3), a .pdf plot of global resource usage (Figure K.5), and a .pdf plot of resource utilization (Figure K.6).

```
Makespan                    : 54 days
Start delay                 : 0 days
Project delay               : 12 days
Combined slack sufficiency : 228
Relative gap                : 28.57%
```

Listing K.3: Performance measures of j3033_3

K.3 Log

A log is presented indicating what each agent is doing (Listing K.4). It can be used for debugging purposes and for analytic purposes, for example to see agent decision making.

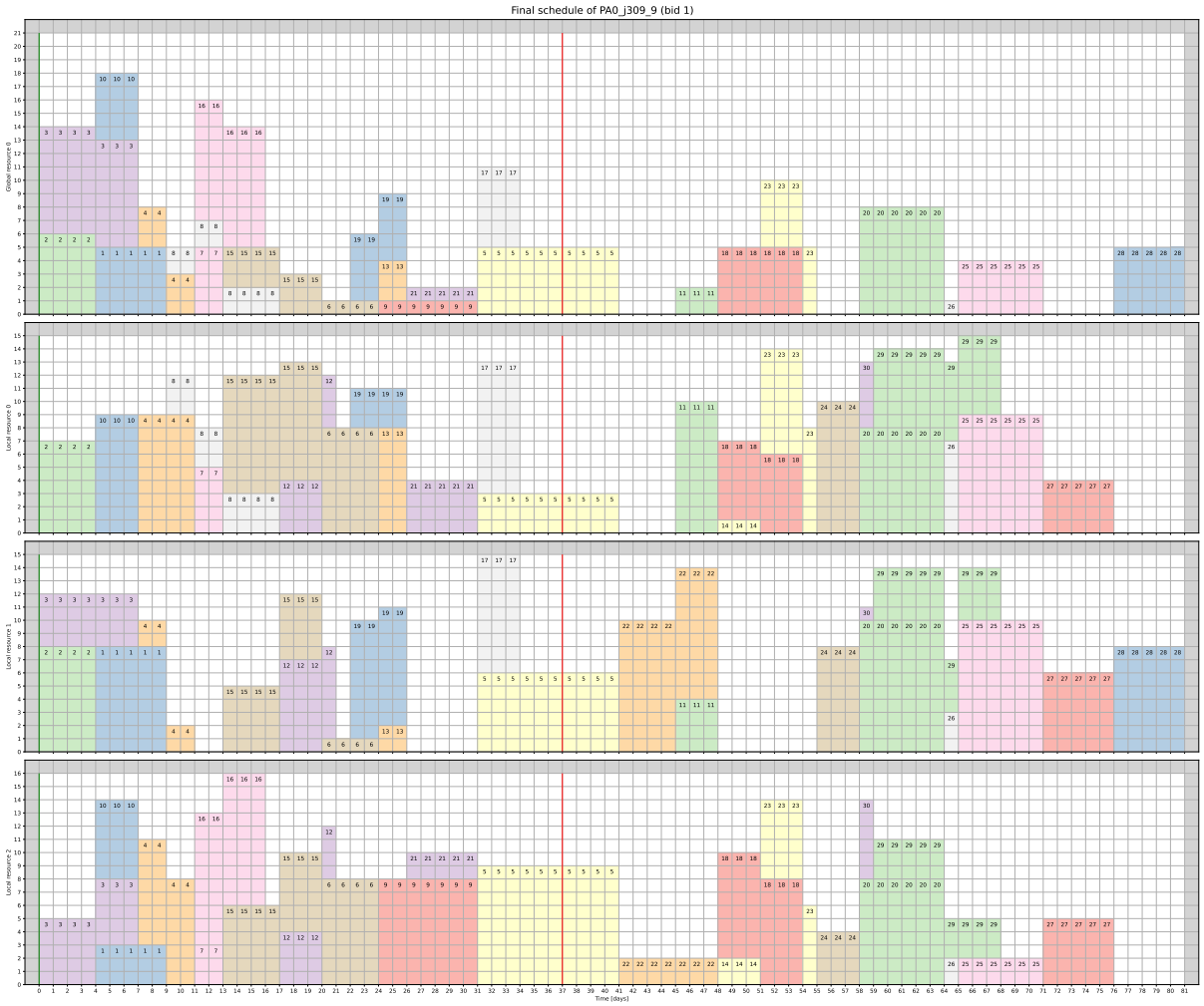


Figure K.3: Final schedule of project instance j309_9

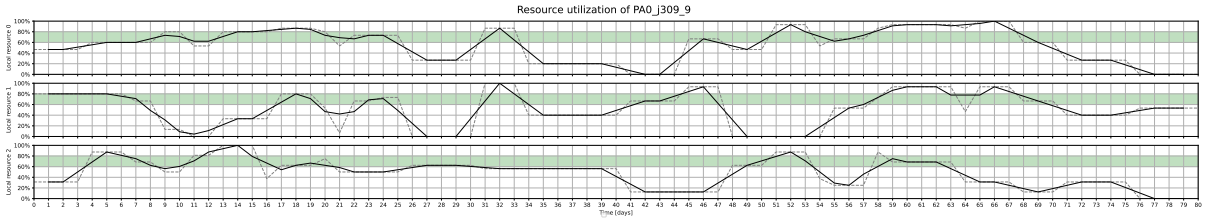


Figure K.4: Resource usage of project instance j309_9

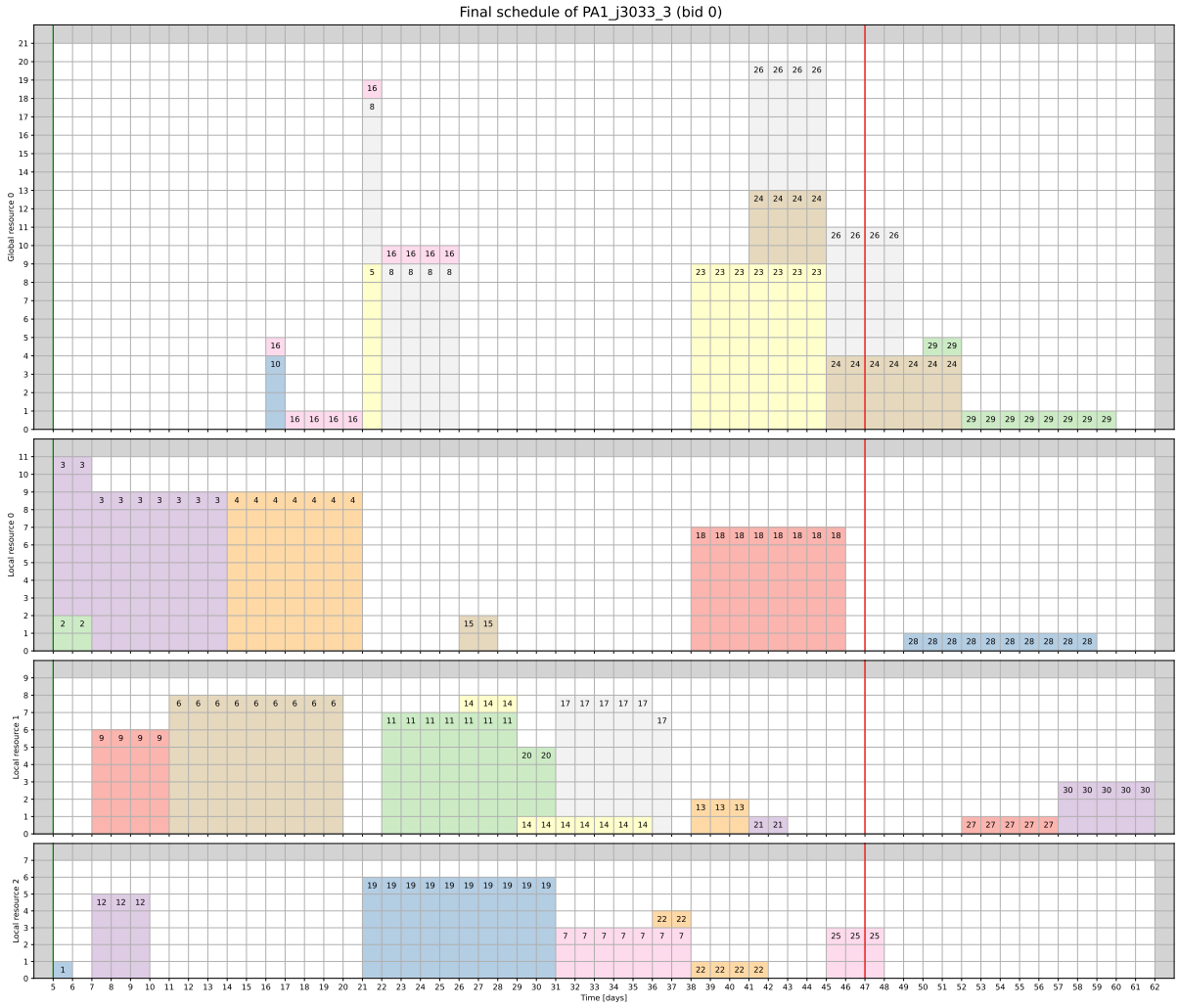


Figure K.5: Final schedule of project instance j3033_3

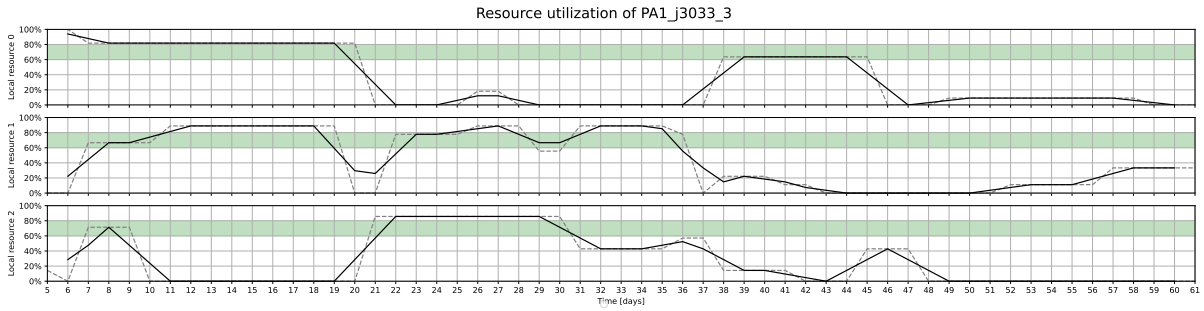


Figure K.6: Resource usage of project instance j3033_3

```
Auction:  ==== START ====
Auction.Auctioneer: Initializing
Auction.Auctioneer: Importing auction
Auction.Auctioneer: Importing auction: resource agent
Auction.ResourceAgent: Initializing
Auction.Auctioneer: Importing auction: project agents
Auction.Auctioneer: Importing auction: project agent PA0_j309_9
Auction.PA0_j309_9: Initializing
Auction.PA0_j309_9: Generating critical schedule
Auction.PA0_j309_9: Plotting critical schedule
Auction.PA0_j309_9: Presenting characteristics
Auction.Auctioneer: Importing auction: project agent PA1_j3033_3
Auction.PA1_j3033_3: Initializing
Auction.PA1_j3033_3: Generating critical schedule
Auction.PA1_j3033_3: Plotting critical schedule
Auction.PA1_j3033_3: Presenting characteristics
Auction.Auctioneer: Importing auction completed
Auction.Auctioneer: Starting auction with 2 bidder(s)
Auction.Auctioneer: 2 bidder(s) bidding in initial phase 0
Auction.PA1_j3033_3: Generating bid 0 by solving bidding problem
Auction.PA1_j3033_3: Schedule has combined slack sufficiency of 214
Auction.Auctioneer: Received bid from PA1_j3033_3 with valuation 9983.0
Auction.PA0_j309_9: Generating bid 0 by solving bidding problem
Auction.PA0_j309_9: Schedule has combined slack sufficiency of 342
Auction.Auctioneer: Received bid from PA0_j309_9 with valuation 9964.0
Auction.Auctioneer: Determining winning bidder(s) by solving WDP
Auction.Auctioneer: Accepting initial bid of PA1_j3033_3 with criteria value 3016.52
Auction.Auctioneer: Rejecting initial bid of PA0_j309_9 with criteria value 2103.94
Auction.Auctioneer: Calculating demand ratio 0
Auction.Auctioneer: 1 bidder(s) bidding in final phase 0
Auction.PA1_j3033_3: Generating modified bid 0 by solving bid modification problem
Auction.Auctioneer: Determining winning bidder(s) by solving WDP
Auction.Auctioneer: Identifying winner: first subtracting resources
Auction.ResourceAgent: Subtracting allocated resources from available resources
Auction.Auctioneer: Updating available global resource level to 1
Auction.Auctioneer: Accepting final bid of PA1_j3033_3 with criteria value 2947.03
Auction.PA1_j3033_3: Presenting output
Auction.Auctioneer: 1 bidder(s) bidding in initial phase 1
Auction.PA0_j309_9: Generating bid 0 by solving bidding problem
Auction.PA0_j309_9: Schedule has combined slack sufficiency of 377
Auction.Auctioneer: Received bid from PA0_j309_9 with valuation 9958.0
Auction.Auctioneer: Determining winning bidder(s) by solving WDP
Auction.Auctioneer: Accepting initial bid of PA0_j309_9 with criteria value 1979.41
Auction.Auctioneer: Calculating demand ratio 1
Auction.Auctioneer: 1 bidder(s) bidding in final phase 1
Auction.PA0_j309_9: Generating modified bid 1 by solving bid modification problem
Auction.Auctioneer: Determining winning bidder(s) by solving WDP
Auction.Auctioneer: Identifying winner: first subtracting resources
Auction.ResourceAgent: Subtracting allocated resources from available resources
Auction.Auctioneer: Updating available global resource level to 2
Auction.Auctioneer: Accepting final bid of PA0_j309_9 with criteria value 1885.79
Auction.PA0_j309_9: Presenting output
Auction.Auctioneer: Closing auction
Auction.Auctioneer: Presenting selected outputs
Auction:  ===== END =====
```

Listing K.4: Log of auction

Appendix L

Population and sample size

In an ideal world, a schedule is subjected to every possible disruption to determine the performance of a project scheduling method. This is not efficient due to the number of possible combinations of disruptions (Table L.1).

Table L.1: Number of combinations of disruptions in a scenario and size of sample

total tasks			nr disruptions	combinations	sample
30·	2 =	60	6	$5,01 \cdot 10^{07}$	300
30·	5 =	150	15	$1,62 \cdot 10^{20}$	750
30·	10 =	300	30	$1,73 \cdot 10^{41}$	750
30·	20 =	600	60	$2,77 \cdot 10^{83}$	750
90·	2 =	180	18	$2,55 \cdot 10^{24}$	900
90·	5 =	450	45	$2,13 \cdot 10^{62}$	900
90·	10 =	900	90	$5,13 \cdot 10^{125}$	900
90·	20 =	1.800	180	$4,20 \cdot 10^{252}$	900
120·	2 =	240	24	$6,54 \cdot 10^{32}$	-
120·	5 =	600	60	$2,77 \cdot 10^{83}$	-
120·	10 =	1.200	120	$1,00 \cdot 10^{168}$	-
120·	20 =	2.400	240	$1,86 \cdot 10^{337}$	-

The first column presents the MPSPLib instances sizes, being the number of tasks per project multiplied by the nr of projects. The second column indicates how many tasks are disrupted, which is 10% of the number of tasks per project. The third column presents the number of all disruption combinations. This indicates why it is infeasible to test all possible instances.

The last column indicates the selected sample size. Subjecting an instance to more scenarios (a higher sample size) increases the precision of the results. The sample sizes are chosen such that the required computational time is 'acceptable' while retaining an 'acceptable' precision. The simulation could be executed until a certain time limit. Additionally, it is possible to estimate the number of runs required to attain an answer within a certain error margin (Equation L.1).

$$n > \left(\frac{Z_{\alpha/2} \cdot \hat{\sigma}}{\epsilon} \right)^2 \tag{L.1}$$

In the formula, $Z_\alpha = 1.96$, being the 95% Confidence Interval (CI), $\hat{\sigma}$ is an estimate for the standard deviation, for example attained in short test runs, and ϵ is the tolerated error margin. Despite these options to terminate the simulation, the author chose fixed numbers for the sample size as they produce an 'acceptable' precision.

To indicate the computational hardness, the sample size is compared to population (being the number of combinations). Even for the first case, which has the smallest population, the sample tests only covers 0,0006% of all possible combinations. Therefore, simulations are required.

Appendix M

Full results

Table M.1: Output of project scheduling with and without robustness modification

	Original					Robust				
Instance	**AMS**	**TMS**	**APD**	**SAPD**	**CT**	**AMS**	**TMS**	**APD**	**SAPD**	**CT**
30_2_1	79.0	93	23.0	19.80	0.652	80.0	91	24.0	15.56	1.160
30_2_2	66.5	79	27.0	21.21	0.394	69.0	81	29.5	20.51	0.679
30_2_3	61.5	67	4.0	5.66	0.208	68.5	77	11.0	9.90	0.441
30_2_4	57.5	68	18.0	18.38	0.494	64.0	77	24.5	21.92	0.743
30_2_5	69.0	90	24.5	34.65	0.455	71.5	93	27.0	35.36	0.742
30_5_1	70.6	104	18.6	17.42	0.460	74.2	109	22.2	15.45	1.320
30_5_2	68.0	109	20.0	18.34	1.293	70.6	113	22.6	18.99	2.142
30_5_3	92.8	150	42.4	22.69	2.495	100.0	166	49.6	28.20	3.535
30_5_4	51.8	76	0.0	0.00	0.629	55.4	81	3.6	1.14	0.783
30_5_5	73.8	112	26.0	14.47	0.826	79.4	126	31.6	19.14	1.739
30_5_6	83.6	134	33.2	36.08	2.245	86.4	151	36.0	34.15	3.672
30_10_1	124.8	217	90.8	60.15	8.527	131.7	231	97.7	63.68	17.205
30_10_2	68.0	132	18.7	18.46	1.669	70.7	132	21.4	17.35	3.265
30_10_3	145.5	286	96.2	76.17	7.834	156.8	300	107.5	81.86	18.869
30_10_4	79.4	187	28.7	35.79	2.885	83.6	198	32.9	37.68	7.624
30_10_5	120.6	221	64.7	55.49	2.046	135.0	240	79.1	68.32	9.957
30_20_1	264.7	486	207.7	134.58	14.401	283.15	523	226.15	147.33	148.849
30_20_2	143.45	317	91.75	80.00	6.774	147.5	344	95.8	80.09	40.921
30_20_3	159.15	340	112.6	90.42	23.938	166.6	356	120.05	95.96	65.687
30_20_4	95.35	225	39.3	52.50	8.123	101.25	246	45.2	57.02	25.193
30_20_5	225.15	470	170.2	125.26	17.863	240.45	487	185.5	131.71	76.405
90_2_1	88.0	88	0.0	0.00	1.378	94.0	94	6.0	0.00	3.548
90_2_2	138.0	151	46.0	33.94	11.923	143.5	158	51.5	36.06	18.357
90_2_3	118.0	141	16.0	15.56	4.385	124.5	150	22.5	19.09	8.833
90_2_4	85.5	92	1.0	1.41	2.881	89.5	96	5.0	2.83	4.866
90_2_5	103.0	128	3.5	4.95	0.332	109.0	135	9.5	6.36	4.221
90_2_5_AC1	185.5	234	101.5	61.52	11.817	196.5	248	112.5	65.76	22.184

Table M.1: Output of project scheduling with and without robustness modification (continued)

	Original					Robust				
Instance	**AMS**	**TMS**	**APD**	**DPD**	**CT**	**AMS**	**TMS**	**APD**	**DPD**	**CT**
90_2_5_AC2	294.5	389	217.0	121.62	19.103	303.0	410	225.5	139.30	39.268
90_2_5_AC3	146.0	202	62.0	74.95	8.238	153.0	211	69.0	77.78	16.732
90_2_5_AC4	301.5	403	227.5	136.47	14.365	308.5	421	234.5	152.03	40.407
90_2_5_AC5	72.5	83	5.5	7.78	3.079	76.5	87	9.5	7.78	5.455
90_2_5_AC6	183.0	229	99.0	57.98	10.672	189.0	233	105.0	55.15	21.130
90_2_5_AC7	298.0	391	220.5	126.57	15.548	302.0	403	224.5	137.89	38.941
90_2_5_AC8	143.0	191	59.0	70.71	9.410	152.0	204	68.0	76.37	17.249
90_2_5_AC9	296.5	388	222.5	129.40	13.716	308.0	415	234.0	151.32	34.330
90_2_5_AC10	110.0	145	43.0	49.50	7.999	116.0	154	49.0	53.74	12.161
90_5_1	79.0	79	0.0	0.00	10.218	83.0	83	4.0	0.00	16.395
90_5_2	104.8	132	13.2	22.52	4.188	109.0	133	17.4	20.82	10.042
90_5_3	98.2	147	7.0	10.75	5.910	104.0	153	12.8	10.99	12.636
90_5_4	107.8	154	18.4	24.05	13.646	114.0	155	24.6	24.76	29.027
90_5_5	117.0	177	31.2	27.44	6.017	119.4	190	33.6	30.83	17.374
90_5_5_AC1	365.0	667	282.4	200.45	45.024	387.0	702	304.4	212.48	187.721
90_5_5_AC2	411.8	758	331.6	261.49	47.362	425.8	792	345.6	270.73	214.465
90_5_5_AC3	218.2	405	135.8	116.77	30.670	224.4	407	142.0	116.65	87.437
90_5_5_AC4	578.0	974	504.0	295.84	46.236	593.2	990	519.2	310.27	301.738
90_5_5_AC5	186.8	310	119.8	90.55	35.736	196.8	333	129.8	99.33	83.824
90_5_5_AC6	371.8	668	289.2	208.81	44.397	401.2	713	318.6	225.44	179.540
90_5_5_AC7	415.2	767	335.0	264.53	47.820	435.2	792	355.0	270.99	219.762
90_5_5_AC8	222.0	397	139.6	118.39	25.598	229.4	415	147.0	122.84	99.032
90_5_5_AC9	588.0	983	514.0	308.38	44.757	608.0	1009	534.0	317.37	263.638
90_5_5_AC10	200.2	323	133.2	96.21	27.886	209.6	332	142.6	99.26	80.774
90_10_1	165.5	226	76.5	34.69	30.647	173.2	264	84.2	40.80	158.706
90_10_2	92.5	150	12.5	12.78	30.873	98.1	161	18.1	14.90	55.943
90_10_3	148.9	266	59.2	65.42	31.767	153.5	259	63.8	63.93	130.417
90_10_4	94.1	151	6.8	10.57	16.673	99.2	157	11.9	11.53	31.915

Table M.1: Output of project scheduling with and without robustness modification (continued)

	Original					Robust				
Instance	**AMS**	**TMS**	**APD**	**DPD**	**CT**	**AMS**	**TMS**	**APD**	**DPD**	**CT**
90_10_5	162.4	305	67.1	67.14	35.477	172.1	313	76.8	63.49	122.216
90_10_5_AC1	371.6	798	287.7	228.95	85.132	392.2	830	308.3	237.75	240.620
90_10_5_AC2	397.3	865	316.7	272.78	105.628	417.9	910	337.3	283.54	265.071
90_10_5_AC3	169.6	323	79.8	81.22	40.985	178.9	336	89.1	83.69	100.156
90_10_5_AC4	405.8	764	331.8	215.80	53.733	431.3	807	357.3	224.82	213.183
90_10_5_AC5	237.1	435	170.1	127.82	71.290	248.5	456	181.5	134.78	152.584
90_10_5_AC6	384.1	786	300.2	233.22	91.699	404.9	840	321.0	247.87	244.488
90_10_5_AC7	229.2	460	148.6	140.81	57.561	245.9	506	165.3	151.53	149.233
90_10_5_AC8	180.9	330	91.1	87.43	50.051	192.8	352	103.0	92.14	103.248
90_10_5_AC9	166.0	270	92.0	67.93	28.043	170.5	275	96.5	69.05	93.740
90_10_5_AC10	141.7	222	74.7	57.09	34.084	146.3	234	79.3	57.53	93.057
90_20_1	103.3	128	6.3	9.98	54.282	110.75	135	13.75	8.88	102.238
90_20_2	106.1	182	12.5	10.66	13.309	111.8	188	18.2	10.84	26.926
90_20_3	96.1	140	10.05	10.53	30.074	101.75	148	15.7	11.30	51.589
90_20_4	125.75	220	38.7	40.73	73.657	131.1	215	44.05	40.29	160.703
90_20_5	143.85	283	53.2	58.76	54.637	147.2	273	56.55	56.86	155.471
90_20_5_AC1	243.1	519	159.2	142.51	134.397	258.55	535	174.65	150.83	417.169
90_20_5_AC2	84.7	140	4.1	9.66	37.628	89.85	152	9.25	11.14	65.484
90_20_5_AC3	112.0	192	22.2	32.64	53.453	118.1	201	28.3	32.72	117.321
90_20_5_AC4	209.85	392	135.85	103.83	69.568	216.7	408	142.7	107.80	349.314
90_20_5_AC5	87.35	165	20.35	29.56	42.256	91.5	163	24.5	28.87	88.289
90_20_5_AC6	157.25	283	73.35	69.42	82.879	159.5	309	75.6	71.60	254.360
90_20_5_AC7	166.45	322	85.85	87.88	103.274	173.75	343	93.15	91.88	292.783
90_20_5_AC8	122.8	200	33.0	40.02	59.167	128.45	218	38.65	41.77	145.729
90_20_5_AC9	260.0	463	186.0	124.43	64.347	271.1	485	197.1	133.12	456.785
90_20_5_AC10	315.0	565	248.0	157.78	172.796	331.25	596	264.25	165.49	664.171
120_2_1	178.5	230	68.5	72.83	2.097	171.5	210	61.5	54.45	7.909
120_2_2	143.5	155	43.5	19.09	15.644	151.0	165	51.0	22.63	21.149

Table M.1: Output of project scheduling with and without robustness modification (continued)

	Original					Robust				
Instance	**AMS**	**TMS**	**APD**	**DPD**	**CT**	**AMS**	**TMS**	**APD**	**DPD**	**CT**
120_2_3	240.5	330	140.5	116.67	15.826	251.0	331	151.0	103.24	31.320
120_2_4	158.5	175	61.5	19.09	20.840	157.0	171	60.0	15.56	24.353
120_2_5	121.5	136	21.0	9.90	1.051	127.0	141	26.5	9.19	5.519
120_2_5_AC1	171.0	260	97.5	119.50	22.751	174.0	262	100.5	118.09	29.886
120_2_5_AC2	122.0	150	33.0	25.46	12.797	133.0	166	44.0	32.53	18.159
120_2_5_AC3	170.0	222	87.5	58.69	15.314	182.0	241	99.5	68.59	23.893
120_2_5_AC4	113.0	141	23.5	33.23	11.444	123.5	157	34.0	41.01	17.220
120_2_5_AC5	104.0	128	26.0	24.04	11.908	109.5	135	31.5	26.16	14.916
120_2_5_AC6	168.5	251	95.0	115.97	19.103	180.0	270	106.5	126.57	29.816
120_2_5_AC7	125.0	156	36.0	29.70	11.984	134.5	169	45.5	34.65	18.915
120_2_5_AC8	174.0	225	91.5	64.35	13.868	177.5	227	95.0	62.23	20.674
120_2_5_AC9	110.5	136	21.0	29.70	10.867	115.5	141	26.0	29.70	13.779
120_2_5_AC10	103.0	126	25.0	22.63	9.709	110.0	136	32.0	26.87	13.677
120_5_1	88.0	88	15.0	0.00	2.637	92.0	92	19.0	0.00	7.509
120_5_2	124.2	201	43.6	44.89	9.087	129.8	208	49.2	45.92	20.010
120_5_3	181.2	244	79.0	39.89	63.014	192.4	257	90.2	44.87	91.070
120_5_4	169.8	246	73.6	51.33	55.788	178.4	260	82.2	54.70	79.507
120_5_5	193.2	304	90.2	77.16	25.767	199.4	322	96.4	79.43	63.239
120_5_5_AC1	332.0	672	251.2	252.21	75.085	365.8	734	285.0	271.06	150.870
120_5_5_AC2	201.2	366	111.0	103.46	52.628	210.2	391	120.0	113.06	82.821
120_5_5_AC3	355.4	633	259.6	194.83	77.101	374.6	669	278.8	208.00	133.975
120_5_5_AC4	227.6	476	146.8	150.84	44.663	238.2	487	157.4	151.60	87.618
120_5_5_AC5	345.8	733	247.6	251.91	58.097	361.4	767	263.2	263.38	139.530
120_5_5_AC6	342.0	681	261.2	249.45	75.152	367.4	737	286.6	273.57	155.552
120_5_5_AC7	206.8	366	116.6	108.14	53.587	224.8	394	134.6	116.44	90.977
120_5_5_AC8	361.8	628	266.0	196.01	79.693	389.6	668	293.8	211.64	159.856
120_5_5_AC9	224.0	441	143.2	148.54	50.619	232.0	469	151.2	157.09	92.455
120_5_5_AC10	350.8	725	252.6	255.18	62.382	368.6	770	270.4	271.02	129.264

Table M.1: Output of project scheduling with and without robustness modification (continued)

	Original					Robust				
Instance	**AMS**	**TMS**	**APD**	**DPD**	**CT**	**AMS**	**TMS**	**APD**	**DPD**	**CT**
120_10_1	114.9	163	49.9	29.39	17.057	118.0	170	53.0	28.46	71.421
120_10_2	176.4	275	85.5	60.89	45.312	185.5	281	94.6	60.45	105.391
120_10_3	116.4	178	17.0	20.99	12.331	121.6	194	22.2	21.91	37.052
120_10_4	319.2	508	222.2	94.02	216.976	332.8	509	235.8	95.74	347.880
120_10_5	316.2	550	207.4	142.65	25.056	333.6	588	224.8	152.21	203.407
120_10_5_AC1	377.9	879	293.2	278.73	166.937	395.2	940	310.5	294.10	447.785
120_10_5_AC2	225.5	459	134.6	123.27	125.164	235.9	491	145.0	130.76	231.386
120_10_5_AC3	284.8	555	184.9	151.91	119.766	298.1	585	198.2	162.41	307.304
120_10_5_AC4	226.3	492	137.0	138.41	118.872	241.8	530	152.5	150.82	264.491
120_10_5_AC5	213.8	492	115.0	129.33	63.501	223.0	533	124.2	139.19	207.004
120_10_5_AC6	219.9	469	129.8	139.14	87.838	228.2	490	138.1	145.45	236.713
120_10_5_AC7	118.8	191	27.9	38.62	56.853	122.5	207	31.6	37.70	104.687
120_10_5_AC8	140.5	230	40.6	45.34	61.441	148.3	248	48.4	48.79	153.026
120_10_5_AC9	118.3	190	29.0	34.62	63.137	121.6	204	32.3	33.46	108.077
120_10_5_AC10	130.5	239	31.7	42.20	55.768	136.7	223	37.9	42.85	94.708
120_20_1	88.15	95	15.15	4.40	19.337	91.55	99	18.55	4.86	74.259
120_20_2	141.75	240	42.4	26.65	94.126	147.2	259	47.85	25.37	181.301
120_20_3	145.9	261	52.7	37.05	131.319	151.4	271	58.2	37.88	205.061
120_20_4	129.75	233	34.3	30.47	120.873	135.4	249	39.95	32.13	191.128
120_20_5	133.05	208	33.55	11.66	70.185	139.25	216	39.75	12.76	110.822
120_20_5_AC1	208.95	443	112.95	120.43	187.420	216.65	445	120.65	122.55	539.656
120_20_5_AC2	167.75	359	74.95	85.00	142.505	175.8	364	83.0	89.87	367.679
120_20_5_AC3	476.4	1060	377.05	309.61	491.457	497.5	1106	398.15	321.03	1662.276
120_20_5_AC4	174.55	363	79.1	94.46	208.842	184.65	373	89.2	103.11	409.122
120_20_5_AC5	193.9	420	95.15	104.47	151.563	203.2	466	104.45	110.17	434.862
120_20_5_AC6	422.95	988	326.95	304.64	339.291	450.95	1046	354.95	324.23	1248.533
120_20_5_AC7	421.25	941	328.45	291.16	340.355	445.95	1005	353.15	308.28	1277.268
120_20_5_AC8	226.8	431	127.45	113.16	162.287	238.5	441	139.15	122.05	653.171

Table M.1: Output of project scheduling with and without robustness modification (continued)

	Original					Robust				
Instance	**AMS**	**TMS**	**APD**	**DPD**	**CT**	**AMS**	**TMS**	**APD**	**DPD**	**CT**
120_20_5_AC9	179.45	357	84.0	92.52	152.810	186.95	383	91.5	95.56	463.573
120_20_5_AC10	201.65	416	102.9	108.99	127.362	213.3	433	114.55	115.48	506.522

Appendix N

Full results simulation

The simulation is executed using the settings in Appendix L that indicate the number of scenarios (runs) and disruptions per scenario. Each scenario is executed twice, once on the original solution not made robust and once on the robust solution. For each instance, the in-time delivery, makespan and stability (or system nervousness) are reported. Since it is a Monte Carlo simulation, a confidence interval is presented using a 95% Confidence Interval (CI).

For each run, the in-time delivery is either 0 (past due date) or 1 (within due date). Therefore, an estimator for the mean $\hat{p}$ is the average value of all runs in an instance and the variance is $\hat{p} \cdot (1 - \hat{p})$

For both the makespan and stability, a CI is constructed based on the Central Limit Theory (CLT). From a sample of size n, a sample mean $\bar{Z}$ and sample variance S^2 (using $n - 1$ degrees of freedom) are determined. Using these, a half-width is constructed (Equation N.1) using $Z_\alpha = 1.96$ for a 95% CI.

$$\text{half-width} = Z_\alpha \sqrt{\frac{S^2}{n}} \tag{N.1}$$

Table N.1: Full results simulations

	Original			Robust		
Instance	In-time delivery CI_{lb} μ CI_{ub}	Makespan CI_{lb} μ CI_{ub}	Stability CI_{lb} μ CI_{ub}	In-time delivery CI_{lb} μ CI_{ub}	Makespan CI_{lb} μ CI_{ub}	Stability CI_{lb} μ CI_{ub}
30_2_1	0,41 0,47 0,53	93,36 93,59 93,82	123,77 133,1 142,44	0,81 0,85 0,89	89,91 90,22 90,54	170,68 180,34 190,01
30_2_2	0,28 0,34 0,39	79,82 80,15 80,48	74,94 83,6 92,26	0,82 0,86 0,9	79,67 79,98 80,29	106,1 117,92 129,73
30_2_3	0,34 0,39 0,45	68,83 69,21 69,58	32,56 37,58 42,61	0,9 0,93 0,96	73,49 73,89 74,29	47,01 53,35 59,7
30_2_4	0,13 0,18 0,22	72,39 72,75 73,12	126,38 137,79 149,2	0,53 0,59 0,64	76,79 77,11 77,43	100,4 110,05 119,71
30_2_5	0,45 0,51 0,57	87,75 88,35 88,95	76,71 86,81 96,91	0,97 0,98 1	88,36 89 89,63	75,42 86,45 97,49
30_5_1	0,31 0,34 0,37	108,11 108,44 108,77	283,49 299,64 315,78	0,4 0,43 0,47	110,68 110,99 111,31	372,54 390,7 408,86
30_5_2	0,41 0,44 0,48	110,11 110,27 110,43	436,16 449,04 461,92	0,93 0,94 0,96	110,4 110,58 110,77	530,63 547,67 564,71
30_5_3	0,32 0,36 0,39	151,65 151,9 152,16	525,66 538,77 551,87	0,86 0,89 0,91	163,11 163,41 163,7	520,76 537,1 553,45
30_5_4	0,79 0,82 0,84	76,17 76,2 76,23	2,95 3,45 3,94	1 1 1	79,75 79,82 79,88	4,43 5,04 5,66
30_5_5	0,41 0,45 0,49	113,65 113,91 114,17	596,74 615,58 634,43	0,91 0,92 0,94	124,07 124,26 124,45	431,14 446,72 462,3
30_5_6	0,79 0,82 0,85	133,62 133,75 133,88	279,16 296,09 313,02	0,94 0,96 0,97	149,95 150,02 150,08	322,94 339,62 356,3
30_10_1	0,04 0,06 0,08	224,3 224,64 224,98	3441,77 3480,85 3519,92	0,07 0,09 0,11	237,93 238,29 238,65	3941,58 3989,44 4037,3
30_10_2	0,22 0,25 0,28	135,09 135,45 135,8	575,04 591,32 607,59	0,43 0,47 0,5	133,61 133,88 134,16	544,44 562,12 579,79
30_10_3	0 0,01 0,02	298,99 299,49 299,99	4393,57 4453,94 4514,32	0 0 0,01	315,99 316,47 316,95	4902,36 4986,61 5070,85
30_10_4	0,33 0,36 0,39	190,77 191,32 191,86	1785,16 1847,11 1909,05	0,56 0,6 0,63	199,68 200,07 200,46	1704,81 1758,75 1812,69
30_10_5	0,02 0,03 0,05	235,12 235,76 236,39	2472,5 2545 2617,5	0,24 0,27 0,3	249,84 250,62 251,41	2252,23 2310,77 2369,31
30_20_1	0 0 0	533,17 533,88 534,6	17366,55 17578,4 17790,25	0 0 0	601,78 602,75 603,71	25096,93 25399,53 25702,12
30_20_2	0 0 0	345,99 346,56 347,14	10559,71 10707,02 10854,32	0 0,01 0,02	364,76 365,46 366,17	11595,66 11754,02 11912,37
30_20_3	0 0 0	361,82 362,28 362,75	10233,86 10334,37 10434,88	0 0 0	373,38 373,89 374,4	11840,25 11965,66 12091,07
30_20_4	0,11 0,13 0,16	233,33 233,84 234,34	4606,59 4717,74 4828,88	0,45 0,49 0,52	248,32 248,68 249,05	3730,85 3827,47 3924,08
30_20_5	0 0 0	512,77 513,78 514,78	16927,96 17124,35 17320,75	0 0 0	523,38 524,13 524,88	17135,05 17345,87 17556,68
90_2_1	0,55 0,58 0,61	88,43 88,47 88,51	26,89 28,84 30,8	1 1 1	91,4 91,55 91,69	50,5 52,75 55
90_2_2	0,66 0,69 0,72	147,89 148,29 148,68	1347,35 1363,93 1380,51	0,92 0,94 0,95	148,9 149,27 149,65	1386,18 1402,92 1419,67
90_2_3	0,96 0,97 0,98	130,96 131,37 131,78	846,76 865,62 884,48	1 1 1	131,8 132,3 132,8	899,86 919,01 938,17
90_2_4	0,5 0,53 0,57	92,93 93,03 93,13	76,88 82,51 88,15	0,94 0,95 0,97	94,34 94,41 94,49	79,19 84,16 89,13
90_2_5	0,24 0,27 0,29	129,51 129,59 129,68	159,63 166,84 174,06	1 1 1	130,36 130,47 130,58	186,27 192,78 199,29
90_2_5_AC1	0,01 0,01 0,02	257,27 257,93 258,59	3303,1 3360,09 3417,08	0 0,01 0,02	271,9 272,48 273,06	3708,73 3771,46 3834,19
90_2_5_AC10	0,48 0,51 0,54	145,5 145,85 146,2	1245,62 1284,56 1323,49	0,92 0,93 0,95	147,29 147,65 148	1078,63 1113,17 1147,71
90_2_5_AC2	0 0 0,01	409,72 410,26 410,8	3684,65 3737,23 3789,82	0 0 0	449,99 450,45 450,91	4805,19 4845,58 4885,96
90_2_5_AC3	0,12 0,14 0,17	207,55 207,89 208,23	1409,29 1433,87 1458,45	0,01 0,02 0,03	222,57 222,98 223,39	1470 1499,4 1528,81
90_2_5_AC4	0,19 0,22 0,25	409,93 410,49 411,05	3054,84 3095,27 3135,71	0 0 0	476,43 477,11 477,79	5319,13 5369,32 5419,5
90_2_5_AC5	0,81 0,84 0,86	81,14 81,51 81,88	324,27 339,29 354,31	0,89 0,91 0,93	82,19 82,6 83	341,28 356,42 371,55

	Original			Robust		
Instance	In-time delivery CI_{lb} μ CI_{ub}	Makespan CI_{lb} μ CI_{ub}	Stability CI_{lb} μ CI_{ub}	In-time delivery CI_{lb} μ CI_{ub}	Makespan CI_{lb} μ CI_{ub}	Stability CI_{lb} μ CI_{ub}
90_2_5_AC6	0,01 0,02 0,03	252,66 253,37 254,08	3487,24 3541,71 3596,18	0 0 0,01	260,66 261,26 261,87	3473,93 3537,12 3600,32
90_2_5_AC7	0 0 0	410,69 411,14 411,59	3396,64 3443,63 3490,62	0 0 0	450,86 451,34 451,81	5119,68 5161,75 5203,83
90_2_5_AC8	0,01 0,02 0,03	203,15 203,61 204,08	1859,17 1898,17 1937,17	0,08 0,1 0,12	214,9 215,48 216,06	2118,26 2155,79 2193,32
90_2_5_AC9	0 0,01 0,01	409,96 410,53 411,1	3744,65 3794,26 3843,87	0 0 0	461,28 462,01 462,74	4906,74 4953,19 4999,64
90_5_1	0,1 0,12 0,14	80,79 80,9 81,01	644,18 684,73 725,28	0,64 0,67 0,7	83,36 83,42 83,48	680,84 722,95 765,07
90_5_2	0,23 0,26 0,29	134,32 134,58 134,85	470,55 491,83 513,12	0,44 0,47 0,51	134,31 134,57 134,83	511,99 532,11 552,22
90_5_3	0,32 0,35 0,38	148,28 148,4 148,52	870,3 891,72 913,15	0,98 0,98 0,99	149,38 149,53 149,67	892,47 912,37 932,26
90_5_4	0,18 0,2 0,23	161,41 161,79 162,18	2329,44 2373,17 2416,91	0,11 0,13 0,15	161,27 161,59 161,9	2392,59 2431,78 2470,96
90_5_5	0,14 0,17 0,19	182,91 183,27 183,63	1774,34 1804,54 1834,75	0,91 0,93 0,94	182,72 183,07 183,43	1936,89 1972,13 2007,38
90_5_5_AC1	0 0 0	729,55 730,36 731,17	20865,49 21078,22 21290,94	0 0 0	792,74 793,7 794,65	26859,54 27085,54 27311,55
90_5_5_AC10	0,31 0,34 0,38	328,35 328,98 329,61	5337,55 5451,81 5566,06	0,18 0,2 0,23	345,05 345,86 346,67	6440,5 6589,09 6737,68
90_5_5_AC2	0 0 0	801,07 802 802,93	14896,93 15072,19 15247,44	0 0 0	873,43 874,43 875,43	24445,83 24697,36 24948,9
90_5_5_AC3	0,01 0,01 0,02	427,41 428,11 428,81	10864,38 11029,83 11195,28	0 0 0	453,34 454,05 454,76	11808,93 11968,96 12128,98
90_5_5_AC4	0 0 0	1022,44 1023,33 1024,21	16239,94 16395,1 16550,26	0 0 0	1135,96 1136,93 1137,9	29821,14 30001,08 30181,03
90_5_5_AC5	0,09 0,11 0,13	320,16 320,72 321,28	6349,49 6462,61 6575,74	0,32 0,35 0,38	337,76 338,39 339,02	6903,03 7048,68 7194,34
90_5_5_AC6	0 0 0	729,2 730,03 730,86	21241,97 21452,21 21662,44	0 0 0	783,67 784,57 785,47	23035,45 23275,22 23514,99
90_5_5_AC7	0 0 0	795,8 796,63 797,45	13517,06 13693,08 13869,09	0 0 0	878,55 879,51 880,47	21392,26 21625,54 21858,82
90_5_5_AC8	0 0 0,01	424,55 425,28 426	9266,36 9413,15 9559,94	0 0 0	453,07 453,76 454,46	10822,97 10980,84 11138,7
90_5_5_AC9	0 0 0,01	1021,44 1022,29 1023,15	15898,72 16036,43 16174,14	0 0 0	1126,29 1127,32 1128,35	26693,28 26846,72 27000,16
90_10_1	0,46 0,49 0,52	225,19 225,77 226,34	9641 9756,14 9871,28	1 1 1	243,67 244,82 245,97	9878,23 10015,02 10151,82
90_10_2	0,46 0,5 0,53	151,02 151,29 151,55	6107,08 6152,51 6197,94	0,98 0,99 1	152,55 152,78 153,01	6056,68 6108,46 6160,24
90_10_3	0,81 0,83 0,86	263,84 264,1 264,35	7238,5 7313,13 7387,76	0,6 0,64 0,67	258,81 259,08 259,36	8715,99 8787,6 8859,21
90_10_4	0,13 0,16 0,18	156,61 156,9 157,19	2026,64 2070,12 2113,59	0,51 0,55 0,58	157,45 157,67 157,89	2297,39 2339,88 2382,37
90_10_5	0,05 0,06 0,08	318,37 318,98 319,59	8199,34 8277,08 8354,82	0,99 1 1	301,94 302,56 303,19	8965,72 9069,82 9173,91
90_10_5_AC1	0 0 0	874,94 875,75 876,56	47411,34 47841 48270,67	0 0 0	944,72 945,67 946,61	61666,44 62193,8 62721,16
90_10_5_AC10	0,74 0,77 0,79	222,58 222,8 223,03	4613,07 4700,56 4788,04	0,8 0,82 0,85	231,62 231,9 232,19	5716,65 5813,57 5910,49
90_10_5_AC2	0 0 0	910,38 911,28 912,18	38857,46 39226,46 39595,46	0 0 0	993,99 994,91 995,83	52645,04 53076,86 53508,69
90_10_5_AC3	0,02 0,03 0,05	340,65 341,32 342	16300,34 16520 16739,66	0,03 0,04 0,05	354,71 355,47 356,23	17036,21 17240,78 17445,34
90_10_5_AC4	0 0 0	797,14 797,81 798,47	21355,45 21509,6 21663,76	0 0 0	857,41 858,05 858,68	29117,46 29292,22 29466,99
90_10_5_AC5	0,02 0,03 0,05	455,81 456,59 457,37	15875,53 16132,77 16390,01	0,05 0,06 0,08	474,52 475,31 476,09	14647,26 14880,88 15114,5
90_10_5_AC6	0 0 0	866,99 867,76 868,53	49498,05 49979,68 50461,3	0 0 0	930,62 931,57 932,52	58397,29 58992,43 59587,58
90_10_5_AC7	0 0 0	483,02 483,58 484,15	19102,41 19264,5 19426,6	0,23 0,26 0,29	510,14 510,6 511,05	19960,19 20131,98 20303,77
90_10_5_AC8	0 0,01 0,01	349,29 349,88 350,46	17425,86 17734,18 18042,5	0,09 0,11 0,14	364,54 365,29 366,04	16381,67 16625,04 16868,4
90_10_5_AC9	0,41 0,44 0,48	271,12 271,28 271,43	5927,13 5998,81 6070,5	0,43 0,47 0,5	276,59 276,84 277,08	6858,58 6937,18 7015,78

	Original			Robust		
Instance	In-time delivery CI_{lb} μ CI_{ub}	Makespan CI_{lb} μ CI_{ub}	Stability CI_{lb} μ CI_{ub}	In-time delivery CI_{lb} μ CI_{ub}	Makespan CI_{lb} μ CI_{ub}	Stability CI_{lb} μ CI_{ub}
90_20_1	0,61 0,64 0,67	128,51 128,59 128,68	8224,21 8320,8 8417,4	1 1 1	129,11 129,37 129,64	8852,56 8948,04 9043,51
90_20_2	0,13 0,15 0,18	185,17 185,42 185,68	2460,48 2495,92 2531,36	0,82 0,84 0,86	185,53 185,78 186,04	2721,96 2755,52 2789,09
90_20_3	0,61 0,64 0,67	138,09 138,34 138,58	5387,34 5463,02 5538,71	1 1 1	138,82 139,1 139,38	6469,69 6550,84 6632
90_20_4	0,37 0,4 0,43	221,61 221,79 221,97	12108,1 12209,16 12310,21	0,51 0,54 0,58	215,87 216,18 216,5	13593,28 13704,1 13814,92
90_20_5	0,73 0,76 0,79	282,17 282,36 282,54	16119,99 16271,02 16422,06	0,2 0,22 0,25	279,28 279,71 280,13	16767,71 16881,93 16996,14
90_20_5_AC1	0,02 0,04 0,05	532,22 532,68 533,13	44142,49 44621,6 45100,7	0 0 0,01	554,33 554,76 555,19	49552,5 49983,16 50413,81
90_20_5_AC10	0,07 0,09 0,11	579,47 580,17 580,87	28676,32 29071,99 29467,65	0,14 0,16 0,18	610,02 610,83 611,64	30350,73 30835,21 31319,69
90_20_5_AC2	0,55 0,58 0,61	140,73 140,94 141,15	7430,28 7521,96 7613,64	1 1 1	144,49 144,81 145,13	7698,08 7783,16 7868,24
90_20_5_AC3	0,69 0,72 0,75	191,35 191,62 191,89	8327,09 8442,36 8557,63	0,89 0,91 0,93	198,35 198,61 198,86	8333,55 8444,85 8556,14
90_20_5_AC4	0,31 0,34 0,38	393,94 394,14 394,34	17963,07 18156,27 18349,47	0,55 0,59 0,62	408,25 408,55 408,84	19217,21 19402,85 19588,49
90_20_5_AC5	0,83 0,85 0,87	165,16 165,28 165,39	4070,39 4141,02 4211,64	0,82 0,84 0,87	162,77 162,97 163,16	4335,28 4411,51 4487,75
90_20_5_AC6	0,57 0,6 0,63	283,14 283,43 283,73	21788,42 22049,57 22310,73	0,99 1 1	303 303,27 303,53	18102,06 18252,31 18402,56
90_20_5_AC7	0,73 0,76 0,79	321,3 321,45 321,6	17795,57 17966,02 18136,48	0,84 0,87 0,89	338,64 338,92 339,21	18022,64 18192,38 18362,12
90_20_5_AC8	0,58 0,61 0,65	200,58 200,86 201,15	12826,68 12976,39 13126,09	0,95 0,96 0,98	210,77 211,15 211,53	13255,28 13410,1 13564,92
90_20_5_AC9	0,08 0,1 0,12	468,41 468,7 469	19484,09 19672,71 19861,33	0,05 0,07 0,09	492,25 492,62 493	21712,94 21891,56 22070,18