QATAR UNIVERSITY

COLLEGE OF ENGINEERING

ROBUST RESOURCE INVESTMENT PROBLEM WITH TIME-DEPENDENT RESOURCE COST AND TARDINESS PENALTY

By

ASEM ADEL HATTAB

A THESIS SUBMITTED TO THE FACULTY OF

COLLEGE OF ENGINEERING

IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE IN

ENGINEERING MANAGEMENT

JUNE 2016

Abstract

The **Resource Investment Problem (RIP)** is a variant of the well-known **Resource Constraint Project Scheduling Problem (RCPSP)** that requires finding the optimal resource allocation, given a preset completion date, with the objective of minimizing the total cost.

The practical relevance of RIP is very obvious; since the decision maker (the project manager for example) wants to know what resources are required to achieve the targeted project completion date. RIP helps to decide the amount of investment in resources that yield the optimal solution, in addition to the optimal tradeoff between completion time and resource investment.

In practice, most of the projects are associated with due dates beyond which a tardiness penalty may be applied. To avoid the tardiness penalty, project managers sometimes decide to add more resources, thereby increasing resource investment cost, to the project to finish earlier.

In this thesis the (RIP) has been extended to consider time-depended resource cost instead of time-independent resource cost in the classical RIP. The problem was named Resource Investment Problem with Time-Dependent Resource Cost and Tardiness Penalty, abbreviated as (RIP-TDRC).

A mathematical model was introduced to simultaneously find the optimal resource assignment and activity staring times. The objective is to minimize the sum of the resources and tardiness cost.

Two versions of this problem are addressed in this thesis: the deterministic version of RIP-TDRC and the stochastic version. For the latter, it is assumed that the activity durations are subject to many uncertainties such as (bad weather conditions, material shortage, employee’s absences …etc.). To solve this problem, a simulation-optimization based algorithm is proposed. This algorithm solves the deterministic problem version iteratively through all possible project completion times and simulates the project considering the uncertainties to find the optimal solution. The performance of the proposed algorithm and the effect of some problem parameters on the solution are assessed through computational experiments.

The experiments revealed the usefulness of the algorithm in finding relatively robust solution for small problem sizes.

TABLE OF CONTENTS

List of figures

List of models

List of tables

List of abbreviations

General Abbreviation

PSP	Project Scheduling Problem
RCPSP	Resource-Constrained Project Scheduling Problem
RIP	Resource Investment Problem
RRP	Resource Renting Problem
RACP	Resource Availability Cost Problem
SGS	Schedule Generation Scheme
N(x,α)	Neighborhood of a solution *x*; set of solutions that can be reached from *x* by applying a simple operation *α*
TS	Tabu Search
SA	Simulated Annealing
GA	Genetic Algorithm
CC/BM	Critical Chain Buffer Management
ADFF	Adapted Float Factor Model
OS	Order Strength
wp	Weighing Parameter
RFDFF	Called Resource Flow-Dependent Float Factor.
RF	Resource Factor
RC	Resource Constrainedness
VADE	Virtual Activity Duration Extension
STC	Starting Time Criticality

Models Abbreviations

$\boldsymbol{AoN}$	activity-on-node
$\boldsymbol{G}$	project scheduling network graph
$\boldsymbol{A}$	Set of project actual activities
$\boldsymbol{V}$	Set of project activities including the dummy ones
$\boldsymbol{n}$	Number of non-dummy activities in set $\boldsymbol{A}$
$\boldsymbol{i}$	Activity index
$\boldsymbol{E}$	set of precedence relations between activities
$\boldsymbol{R}$	Set of renewable resources
$\boldsymbol{m}$	The Number of renewable resources
$\boldsymbol{k}$	Resource index
$\boldsymbol{B_k}$	Availability of resource $\boldsymbol{k}$
$\boldsymbol{C_k}$	Cost per unit of resource $\boldsymbol{k}$
$\hat{C}_{\boldsymbol{k}}$	Cost per unit of resource $\boldsymbol{k}$ per unit of time
$\boldsymbol{p_i}$	Duration of activity $\boldsymbol{i}$
$\boldsymbol{b_{ik}}$	Requirement of resource k by activity $\boldsymbol{i}$
Δ	The project due date
$\boldsymbol{T}$	Project Tardiness
$\boldsymbol{x_{it}}$	binary variables, where $\boldsymbol{x_{it}} = 1$, if activity i starts at time period $\boldsymbol{t}$ and $\boldsymbol{x_{it}} = 0$ otherwise
$\boldsymbol{t}$	*Time index*
$\boldsymbol{ES_i}$	Earliest possible start time for activity $\boldsymbol{i}$
$\boldsymbol{LS_i}$	latest possible start time for activity i

$\boldsymbol{H}$	is the scheduling horizon starts a t = 0 to $\boldsymbol{LS}_{n+1}$
$\boldsymbol{IS}$	Incompatible set
$\boldsymbol{K}$	Total resource availability cost in a Project
$\boldsymbol{T}$	Total tardiness cost in a Project
$\boldsymbol{Z}$	Total project cost
$\Theta_{\boldsymbol{min}}$	Minimum project completion time
$\Theta_{\boldsymbol{max}}$	Maximum project completion time
$\Theta_{\boldsymbol{opt}}$	Project completion time corresponding to the deterministic optimal solution
$\boldsymbol{Z}_{\boldsymbol{opt}}$	Total project cost corresponding to the deterministic optimal solution
$\hat{\Theta}_{opt}$	Planned (Recommended) project completion time corresponding to the optimal solution found by the simulation algorithm for the stochastic problem version
$\acute{Z}_{opt}$	The realized project cost found by the simulation algorithm for the stochastic problem version, it corresponds to the schedule found by imposing completion time to $\hat{\Theta}_{opt}$
$\Theta_{\boldsymbol{act}}$	Realized project completion time after simulation
$\boldsymbol{Z}_{\boldsymbol{act}}$	Realized Project Cost found by Simulation if the project was scheduled based on the optimal solution for the deterministic problem version, related to $\Theta_{\boldsymbol{opt}}$

Acknowledgments

I would like first to thank Prof. Mohamed Haouari for his continues support and direction to complete my thesis.

I would like also to thank my brother Mohamed Hattab for his valuable contribution on writing the required computer codes to complete this thesis.

Dedication

In the memory of my dear grandmothers. For their love, care and prayers. May Allah give them mercy and forgiveness.

1 Introduction

Projects are present everywhere, almost in every field, whether in business, social life or even fun projects. (Schwindt & Zimmermann, 2015).

A Project could mean different things to different people (Demeulemeester & Herroelen, 2002). ISO 8402 has defined the **project** as: "unique process, consisting of a set of coordinated and controlled activities with start and finish dates, undertaken to achieve an objective conforming to specific requirements, including constraints of time, cost and resources" (Demeulemeester & Herroelen, 2002). PMI has defined the project as "a temporary endeavor undertaken to create a unique product, service or result" (PMI, 2013).

Although projects are found in different forms, they share some common attributes. Some of these attributes are (Schwindt & Zimmermann, 2015)

- Consists of series of related activities.
- Having an objective, such as time, budget and specifications.
- Having defined starting and ending time.
- Requires resources.

1.1 Project Management

In today's competitive environment it is essential to deliver quality products within time and budget. Projects are not an exception to this fact, so it was not surprising to see that project management became a hot topic.

Project management is defined according to (PMI, 2013) as "the application of knowledge, skills, tools, and techniques to project activities to meet the project requirements".

Project management deals with the coordination of initiating, planning, decision, execution, monitoring, control, and closing processes in the course of a project (Schwindt & Zimmermann, 2015).

1.2 Project Scheduling

Project scheduling is part of the project planning process. It is defined as the process of determining the timing, duration and resources allocation for each project activity (PMI, 2013). Scheduling provides guidance and pathway for a project to run. It defines certain milestones and deliverables which need to be achieved for successful completion of a project.

Due to the great importance of the project scheduling, project managers have given a lot of attention on finding the most efficient and effective project schedules. This has attracted the scientists to discover methods and techniques to find such schedules. From here the ***Project scheduling problems*** **(PSPs)** have received considerable attention in the operation research field. PSPs usually address two matters: resource and time.

Resources

In project scheduling the term resources can be used to refer either to renewable resources (which can be reused after completing the activity they assigned for) such as (manpower, machinery, staff...etc.) or non-renewable resources_(which are consumed by the activity they are assigned for) such as (materials, money, fuel …etc.).

In this thesis the word resources will be only used to refer to renewable resources.

Referring to the fact that resources are always limited, the problem was then extended to ***Resource-Constrained Project Scheduling Problem*** **(RCPSP)** (Xiong, Chen, Yang, Zhao, & Xing, 2012) RCPSP is mainly concerned with allocating the limited resources to the project activities taking into consideration the precedence constraints.

The importance of RCPSP in various fields such as (construction, public infrastructure, product and process design, implementation of communication systems, research and development (R&D), software development, maintenance operations...etc.,) has attracted the attention of researchers and project managers (Chtourou & Haouari, 2008) and called for more emphasis on finding good solutions for this problem.

However, in the practical life most often there is a way to add more resources if needed by the project. For example, a construction company with limited number of employees can hire or rent additional employees to expedite work, or more machines can be rented from external source to be added to the available ones, to be able to schedule more activities in parallel. Based on this fact **Resource Investment Problem (RIP)** as an extension to RCPSP has gained some attention.

RIP is concerned with finding the optimal resource combination to achieve some project objective (usually minimizing resource cost). In contrary to RCPSP, RIP main objective is not to find the minimum project duration.

The practical relevance of RIP is very obvious. For example, the decision maker (the project manager) wants to decide about the resources required to achieve the targeted project completion date, budget or any other project objective.

RIP helps to make a decision about the amount of investment in resource to yield the optimal solution, and the optimal tradeoff between time and resource investment.

The traditional Resource Investment Problem assumes that the resources are time-independent. In other words, the per unit availability cost of resource is fixed for the whole project period, whether the resource is used for one-time unit or for all the project life. This is very similar to the assumption that all the resources are purchased for the project only.

In this thesis the concept of resource availability cost will be extended to discuss time-dependent resource cost. So, the per unit cost of resource will be replaced by cost per unit of time per unit of resource. The new problem is named Resource Investment Problem with Time-Dependent Resource Cost (RIP-TDRC).

It is worth mentioning that this problem differs from Resource Renting Problem (RRP). Since the later starts from a given schedule and aims to find optimal renting policy that minimizes the resource cost, while the objective in our problem is to find a schedule that yields the optimal solution. So, finding the activities starting times are part of the problem decision variables in addition to the amount of resources to be assigned for the project.

The differences between the traditional RIP, RRP and RIP-TDRC will be discussed in more details in chapters 2 and 3.

1.3 Uncertainties:

Traditional project scheduling assumes that the parameters of the project schedule are deterministic and not subject to change during the project execution stage. However, this is not the case in actual practice.

Generally, project managers and planners have to work in an environment full of uncertainties. Construction projects for example are subject to continues disruptions caused by accidents, resource breakdowns, worker absenteeism, bad weather conditions, unreliable deliveries of materials...etc. So, the baseline schedules prepared at the beginning of the project will rarely remain feasible in practice (Artigues, Leus, & Nobibon, 2013) (Al-Fawzan & Haouari, 2005).

In the light of the above it is understood that the need of robust schedules is crucial. The ***robust schedule*** is the schedule which is invulnerable to little variation in activity durations caused by uncertainties (Chtourou & Haouari, 2008). Such robust schedules that meet the other project constraints (i.e. resource and precedence constraints) will provide solution for executing the projects under uncertainties and result in avoiding rescheduling the project during execution.

The above reasons inspire the selection of this thesis topic. The main objective of the proposed thesis is to find a feasible solution to RIP-TDRC that is robust and immune from changes in activity duration. This will help project managers in finding good project schedules, and good tradeoff between resources cost and project duration in order minimize the project overall cost.

The goal shall be achieved without sacrificing the schedule robustness and stability to make the produced schedule useable during the project life without the need of rescheduling due to changes in activity durations.

The remainder of this thesis is organized as follows; a comprehensive Literature Review of RCPSP, robust RCPSP and RIP will be presented in chapter 2. In chapter 3, The Resource Investment Problem with Time-Dependent Resource Cost (RIP-TDRC) will be defined. A deterministic exact solution will be introduced. Then the exact algorithm will be combined with simulation to find a robust solution for our problem.

Computational experiment to evaluate the proposed solution strategy will be conducted and explained in chapter 4. Further discussion on the experiment results can be found in chapter 5. Finally, this thesis will be concluded in chapter 6.

2 Literature Review

In this chapter a brief Literature Review of The Resource-Constrained Project Scheduling Problem (RCPSP), Robust RCPSP and Resource Investment Problem will be discussed. The start will be from RCPSP since the thesis topic is considered as a special case of our problem. Also the algorithms and formulations used to solve RCPSP could be used as a base to solve other project scheduling problems.

2.1 Resource Constrained Project Scheduling Problem RCPSP

The main objective of RCPSP is to minimize the project makespan while respecting the precedence and resource constraints. RCPSP is defined below.

RCPSP Problem Definition

In order to define the Resource Constrained Project Scheduling Problem, we assume the graph $\boldsymbol{G}$ represents the activity-on-node (AoN) representation of project scheduling networks, graph $\boldsymbol{G}$ is acyclic (a graph without any directed cycles). Let $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V} = \boldsymbol{A} \cup \{\boldsymbol{0}, \boldsymbol{n}+\boldsymbol{1}\}$ and $\boldsymbol{A} = \{1,\ldots n\}$ is the set of $\boldsymbol{n}$ actual activities (vertices) (activities 0 and $\boldsymbol{n}$+1 are dummy activities represents the start and end of the project respectively), and $\boldsymbol{E}$ is the set of precedence relations between activities (edges). There are $\boldsymbol{m}$ types of renewable resources ($\boldsymbol{k}$ =1….$\boldsymbol{m}$). A constant amount of $\boldsymbol{B_k}$ units of resource $\boldsymbol{k}$ are continuously available from time zero till the end of the project. The duration of activity $\boldsymbol{i} \in \boldsymbol{A}$ is $\boldsymbol{p_i}$ units of time. During this time period a constant amount of $\boldsymbol{b_{ik}}$ units of resource $\boldsymbol{k}$ are occupied (assigned to activity $\boldsymbol{i}$). The project due date is $\boldsymbol{\Delta}$. Preemption is not allowed and all activities have one single execution mode.

The objective of the RCPSP is to find a schedule (determine starting or finishing times for the project activities) that minimizes the total makespan. while the precedence and resource constraints are satisfied.

2.1.1 The Deterministic Resource Constrained Project Scheduling Problem (RCPSP)

The deterministic version of RCPSP assumes no uncertainties at all, neither in the project duration nor resource requirements. The literature is rich of articles in this problem and tried to find optimal and near optimal solutions. The sections below discuss different methods of solving the deterministic RCPSP.

2.1.1.1 Solving the deterministic version of RCPSP

Most of the literature on RCPSP attempts to find an optimum schedule that yields the minimum project duration, under the assumption of certain activity durations. (Leus, 2003). The approaches of solving RCPSP in the literature vary from exact approaches (such as linear programming and branch and bound algorithms) and heuristic approaches (constructive and improvement) (Demeulemeester & Herroelen, 2002).

Exact solutions for RCPSP

There are two common exact methods introduced by scholars in the project scheduling field for solving RCPSP, the linear programming solutions and the branch and bound schemes.

Linear programming based approaches

The formulation proposed by (Pritsker, Watters, & P.Wolfe, 1969), is one of the earliest exact formulations for RCPSP.

The authors suggested $\boldsymbol{x_{it}}$ binary varaibles, where $\boldsymbol{x_{it}}$ ***= 1***, if activity ***i*** starts at time period ***t*** and $\boldsymbol{x_{it}}$***= 0*** otherwise. The mathematical model can be written as follows:

Model 1: RCPSP Mathmatical Formulation

The objective function

$$\mathbf{Minimize} \sum_{t=ES_i}^{LS_i} t.\, x_{n+1,t} \qquad 1$$

Subject to the following constraints

$$\sum_{t=ES_j}^{LS_i} t.\, x_{jt} \geq \sum_{t=ES_i}^{LS_i} t.\, x_{it} + p_i \quad \forall\, (i,j) \in E \qquad 2$$

$$\sum_{i=1}^{n} \sum_{\tau=\max(ES_i, t-p_i+1)}^{\min(LS_i, t)} b_{ik}\, .x_{i\tau} \leq B_k \qquad \forall\, t \in H, \quad \forall\, k \in R \qquad 3$$

Where, H is the scheduling horizon starts a t = 0 to $\boldsymbol{LS}_{n+1}$ and R is the set of renewable resources.

$$\sum_{t=ES_i}^{LS_i} x_{it} = 1 \qquad \forall\, i \in A \cup \{n+1\} \qquad 4$$

$$x_{00} = 1 \qquad 5$$

$$x_{it} = 0 \qquad \forall\, i \in A \cup \{n+1\}, \quad t \in H \backslash \{ES_i, LS_i\} \qquad 6$$

$\boldsymbol{x_{it}} \in \{\mathbf{0}, \mathbf{1}\} \qquad \forall\, \boldsymbol{i} \in \boldsymbol{A} \cup \{\boldsymbol{n}+\mathbf{1}\}, \quad \boldsymbol{t} \in \{\boldsymbol{ES_i}, \boldsymbol{LS_i}\}$ 7

We remark that constraints (2) and (3) are the precedence constraints and resource constraints respectively.

Constraints (4) and (7) impose non-preemption of the project activities. While $\boldsymbol{ES_i}$ stands for the earliest start time, and $\boldsymbol{LS}_i$ is the latest possible start time for any activity.

Constraints

($\boldsymbol{x_{it}} = \mathbf{0} \qquad \forall\, \boldsymbol{i} \in \boldsymbol{A} \cup \{\boldsymbol{n}+\mathbf{1}\}, \quad \boldsymbol{t} \in \boldsymbol{H} \backslash \{\boldsymbol{ES_i}, \boldsymbol{LS_i}\}$ 6)

mean that the start time of any activity can only occur between its earliest start time and its latest start time.

Note that the formulation requires $\boldsymbol{nt_{n+1}}$ decision variables.

(Kaplan,1988) introduced a different formulation for RCPSP. In his formulation the binary decision variables are set to 1 if activity ***i*** is in progress in time period ***t*** and to 0 otherwise. Again this formulation requires $\boldsymbol{nt_{n+1}}$ decision variables (Demeulemeester & Herroelen, 2002).

(Alvarez-Valdés & Tamarit, 1993) presented another exact formulation. Their formulation was based on defining a set ***(IS)*** consists of all ***minimal resource incompatible sets S. Resource incompatible set*** is defined as a set of activities that has no precedence relation in between and violate the resource constraints, if scheduled simultaneously. This set is called ***minimal resource incompatible set*** if it is impossible to remove any activity from that set and still be resource incompatible.

(Mingozzi, Maniezzo, Ricciardelli, & Bianco, 1988) solved RCPSP using a different linear programming formulation. Their formulation was built based on the concept of ***feasible subsets*** (a set that contains activities if scheduled in parallel the precedence and resource constraints are not violated).

(Kone´, Artigues, Lopez, & Mongeau, 2011) proposed two new event based MILP mathematical formulation to solve RCPSP indexed by event instead of time. The first one was based on the start/end formulation where the events correspond to start or end times of the project activities. While the other model was based on (On/Off) event formulation where the events correspond to activity processing times (i.e. the decision variable related to any activity $\boldsymbol{x_{ie}}$ is 1 if activity $\boldsymbol{i}$ is in progress.

The advantage of (Kone´, Artigues, Lopez, & Mongeau, 2011) new formulation is the involvement of less number of variables comparing to other time based formulation and this reason gives advantage for event based formulation in instances with large scheduling horizon.

Branch and bound approaches

Except for heuristic methods, ***branch and bound*** algorithms are considered as the most widely used methods for solving ***RCPSP***. Since ***RCPSP*** belongs to the set of complex combinatorial problems then branch and bound might be the only way which could yield optimal solutions within an acceptable computational time. (Demeulemeester & Herroelen, 2002)

Many branch and bound approaches have been introduced by researchers in project scheduling field. (Demeulemeester & Herroelen, 2002) have published an extensive literature review on those approaches and explained each of them in detail. The following paragraphs summarize their work.

One of the earliest branch and bound techniques is the one offered by (Stinson et al. 1978) named the ***extension alternatives.*** In this approach, sets of the schedulable activities (i.e., their predecessors are completed and do not violate the resource constraint if scheduled simultaneously), are scheduled overtime till a schedule is completed. These sets of activities are called *extension alternatives*. By evaluating all the extension alternatives at each level of the branch and bound tree, the best complete schedule found represents the optimal solution.

An opposite approach to the extension alternatives is the ***minimal delaying alternatives***. In this approach activities are being scheduled until a resource conflict is found. Then at that time instant several alternatives are considered to resolve the resource conflict. In order to resolve a resource conflict one or more activities are delayed to free the resources required for the remaining activities. The optimal schedule can be found by enumerating all the delay alternatives at every branch and bound tree.

Another branch and bound approach to be discussed is what is called ***the minimal forbidden sets***. The ***forbidden set*** is defined as the set of activities that could be scheduled in parallel, but if performed concurrently, the resource constraint is violated. While the minimal forbidden sets, are the ones that do not contain other forbidden set as a subset. To find a feasible schedule each minimal forbidden set is broken (by adding a precedence relation between at least two of its activities) in order to be prevented from running in parallel in the final schedule. The optimal schedule can be found when all feasible schedules are considered.

Schedule scheme approach is based on the idea that for any pair of activities ***i*** and ***j***, there can be only three scheduling options (***i*** precedes ***j***, ***j*** precedes ***i*** or they may overlap with each other). At every level of the branch and bound tree a choice should be made between the three options, for the activities which do not have precedence relation originally.

Float splitting _ heads and tails is a totally different branch and bound approach where the branching is based on the total float of the activities.

Heuristic approaches

Since RCPSP is considered as NP-Hard problem, it is not always feasible to solve the problem using an exact method, _especially for large problem size where solving the problem optimally requires long computation time. In order to resolve this issue, heuristics could be used to find a good feasible solution for such kind of problems.

Heuristics used to solve RCPSP can be categorized under two main categories, (constructive heuristics and improvement heuristics such as meta-heuristics). The first type of heuristics starts from an empty schedule and iteratively adds activities until a feasible schedule is obtained. While the second type (improvement heuristics) starts from a feasible schedule and work on it to find a better solution (Demeulemeester & Herroelen, 2002).

Constructive heuristics

The two major components forming the construction heuristics are the ***scheduling scheme***, which defines the way of constructing the schedule. The second component is the ***priority rule***, which is used as the base to decide the next activity(s) to be scheduled during the scheduling process (Demeulemeester & Herroelen, 2002).

Scheduling Scheme

There are two types of ***schedule generation scheme (SGS), serial schedule generation scheme*** and ***parallel schedule generation scheme***. The *serial schedule generation scheme* (founded by Kelley in 1963) generates feasible schedules by adding activities _one at a time_ till a complete schedule is generated. The algorithm selects the next activity based on the predefined priority list (The list is generated based on a priority rule(s) as discussed later) and selects the earliest possible start time without violating the constraints (precedence or resource) (Kolisch & Hartmann, 1999).

On the other hand, the parallel schedule generation scheme (by brooks and white 1965) iterates based on time incrimination, every time an activity is completed ***(schedule time)*** the balance unscheduled eligible activities are considered for scheduling based on the priority list. The activity with the highest priority is scheduled until there is no eligible activities lift. Then the algorithm moves to the next schedule time and repeat the same procedure until all activities are scheduled (Hartmann & Kolisch, 2000).

(Kolisch, 1996) has reviewed the two scheduling schemes and concluded that both methods are able to produce optimal schedules for PSPs without resource constraint.

However, when extending the problem to include the resource constraint, the author was able to prove that the serial SGS generates ***active schedules*** (a schedule where no activity could be started earlier without delaying another activity or violating a precedence constraint) that includes one optimal solution or more. While the parallel SGS, on the other hand generates ***non delay schedules*** (a schedule where no resource is kept idle at a time when it could begin processing some other activity(s)), which is a subset of the active schedules and don’t guarantee to include an optimal solution, since parallel SGS searches a smaller solution space (Magalhães-Mendes, 2011).

Priority rules

A priority rule is a mapping process which assigns to each activity (in the eligible activity set) a value and an objective stating whether the activity (with the minimum or the maximum value) shall be selected to be scheduled next. To break the tie (if exist) a tie breaking rule shall be applied (Kolisch & Hartmann, 1999).

The priority rules can be classified in five major categories, based on the type of information required to calculate the priority list. These five categories are summarized in Table 1**Error! Reference source not found.** below with examples rom each category (Demeulemeester & Herroelen, 2002):

Table 1: Priority rules categories and examples

Category	**Rule**	**Abbreviation**
Activity based priority rules	Shortest processing time	SPT
	Longest processing time	LPT
Network based priority rules	Most immediate successors	MIS
	Most total successors	MTS
	Least non-related jobs	LNRJ
	Greatest rank positional weight	GRPW
Critical path based priority rules	Earliest start time	EST
	Earliest finish time	EFT
	Latest start time	LST
	Latest finish time	LFT
	Minimum slack	MSLK
	Dynamic earliest start time	ESTD
	Dynamic earliest finish time	EFTD
	Dynamic minimum slack	MSLKD
Resource based priority rules	Greatest resource demand	GRD
	Greatest cumulative resource demand	GCUMRD
	Resource equivalent duration	RED
	Cumulative resource equivalent	CUMRED
Composite priority rules	Weighted resource utilization and	WRUP
	Improved resource scheduling method	IRSM
	Worst case slack	WCS

Proposed Methods

The following methods are the most common methods used to generate schedules using a constructive heuristic.

1. Single pass method

Single pass method employs one SGS and one priority rule to produce one feasible schedule. It is considered as the oldest heuristics approaches to solve RCPSP (Kolisch & Hartmann, 1999).

2. Mutlipass methods

In the Multi pass method, different scheduling schemes may be combined with different priority rules in various ways to find different heuristic solutions, to choose the one which is considered the best. One type of the Mutlipass methods is ***multi scheduling scheme methods.*** Such methods are created by combining the two scheduling schemes (serial and parallel) and the three directions of scheduling (forward, backward & bidirectional), so a total of six combinations can be made. Another type of the multi pass methods is ***Multi priority rule methods*** in such methods multiple priority rules are used with one scheduling scheme and one scheduling direction, each time a different rule is used and the best schedule is chosen (Demeulemeester & Herroelen, 2002).

3. Sampling methods

Sampling usually uses one scheduling scheme and one priority rule to obtain different schedules. The same can be done by determining next activity in the schedule probabilistically based on the priority value of the activities in the eligible set. This probability ***Pi*** (The probability that activity ***i*** will be selected from the schedulable set) can be calculated in many different ways. One of these ways is ***Random sampling***, where the probabilities for each activity in the eligible set are equal. Another way is the ***biased random sampling***; this method biases the probabilities as a function of the priority values of the activities in the eligible set. While in the ***regret based biased random sampling*** the probabilities are calculated indirectly using regret values (Kolisch, 1996).

Improvement heuristics

In contrary to the constructive heuristics, ***improvement heuristics*** start from a feasible schedule, obtained earlier by a constructive heuristic, and improve on that schedule in order to find an improved solution.

To proceed further in discussing the improvement heuristics it is important to understand the ***Neighborhood*** concept. A neighborhood ***N(x,α)*** of a solution ***x*** is defined as a set of solutions that can be reached from ***x*** by applying a simple operation ***α.*** Then it is also important to distinguish between ***unary operators*** that convert one solution to a new solution, and ***binary operators*** that create one or more new solutions from two current solutions (Demeulemeester & Herroelen, 2002).

Descent approaches

The concept of descent approaches is to find a series of new improved solutions. This series of solution is continued as long as the neighborhood of the last current solution contains an improved solution. Descent solutions are categorized to,

Steepest descent that evaluates all the solutions in the neighborhood of the current solution and selects the best one, then continues until no improved solution can be found in the neighborhood of the current solution.

Fastest descent that evaluates all the solutions in the neighborhood of the current solution in random manner, once a better solution is found this solution will be added to series of solutions and the process is repeated until no improved solution can be found in the neighborhood of the current solution.

And iterated descent, since steepest descent and the fastest descent approaches are both terminated in a local optimal solution (found in the neighborhood of the initial solution), so both approaches are considered very sensitive to the that initial solution. The idea of the iterated descent approach is to extend the fastest and steepest descent approaches by adding a random restart procedure to generate new initial solution by using a constructive heuristic. Upon this solution the fastest or steepest descent approach can be restarted. Applying such random restarts will definitely lead to better results (Demeulemeester & Herroelen, 2002).

Metaheuristic approaches

Metaheuristics have been developed by scholars in operations research field to avoid the ***cycling*** phenomenon (revisiting the same local optimal solution repeatedly). To do so different metaheuristics were developed such as ***tabu search, simulated annealing and genetic algorithms.***

Tabu search was developed by (Glover, 1989) to guide the heuristics to continue searching for an improved solution without being confounded by the absence of improving moves, and without falling back into a local optimum from which it previously emerged. To avoid this problem, a tabu list is created to forbid those neighborhood moves that may lead back to a recently visited solution. Typically, such tabu status is overrun if the corresponding neighborhood move would lead to a new overall best solution (aspiration criterion) (Kolisch & Hartmann, 1999).

The concept of tabu search was utilized to create heuristics for solving RCPSP such as the one proposed by (Lee & Kim, 1996). The moves suggested in this algorithm were based on interchanging the priority of two activities randomly. (Baar, Brucker, &

Knust, 1998) introduced two tabu search algorithms to solve RCPSP with the objective of minimizing the makespan. The first algorithm used SGS _with different priority rules_ to find the initial solution. Then the tabu search was applied to the neighborhood based on the list of activities and the elimination of critical arcs. This algorithm shall be terminated after 250 iterations without finding an improved solution. The other algorithm was based on the schedule scheme representation and four types of neighborhood moves. The stopping criterion is similar to the one used in the first algorithm.

Simulated annealing approach was proposed by (Kirkpatrick, Gelatt, & Vecchi, 1983) after realizing the useful connection between statistical mechanics and combinatorial optimization.

The *simulated annealing* approach emphasis on searching for the optimal solution _which can be in any place in the feasible region, rather than finding local optimal solutions. The same can be done by taking random steps to be able to explore as much as possible from the feasible region (Hillier & Lieberman, 2008).

As all search heuristics, the simulated annealing starts with an initial solution, then a new candidate solution is picked from the neighbor of the current solution randomly. The candidate will be accepted if it is better than the current solution, if not it will be accepted or rejected based on a probability usually denoted by the function $e^{\Delta/t}$, where Δ is the difference between of the objective function of the candidate solution and the current solution. While ***t*** is a parameter called the ***temperature,*** (a parameter that measures the tendency to accept the current candidate if it is not improving the current solution) (Hillier & Lieberman, 2008). Usually the temperature starts with relatively large value and then reduced gradually during the simulated annealing process. Selecting large *temperature* value at the beginning leads the probability of solution acceptance to be high, which enables the search to proceed in almost random directions. Reducing the *temperature* value on the other hand, will reduce the probability of accepting an unimproved solution. The search algorithm can be terminated based on a predetermined role such as specified run time or in case if no candidate solution is accepted at any iteration.

(Lee & Kim, 1996) proposed a simulated annealing based heuristic to solve RCPSP. The algorithm was based on priority list representation and the search was based on activity interchange method. The algorithm was stopped after predefined number of unimproved moves.

(Cho & Kim, 1997) extended the research of (Lee & Kim, 1996) to develop a priority scheduling based search heuristic. In which delay schedules as well as non-delay schedules are considered. The same was achieved by using a priority scheduling method for schedule generation with allowing delaying some activities intentionally based on a predefined rules and conditions. As search procedure the authors used the activities interchange method similar to (Lee & Kim, 1996).

The ***genetic algorithm*** **(GA)** developed by Holland in 1975, just like simulated annealing is based on the analogy with a natural phenomenon. This time it is the biological theory of evolution (Hillier & Lieberman, 2008). The genetic algorithm starts with a set of solutions (current population) created earlier by any solution method. Afterwards new solutions are created from the original set of solutions by combining two of them together. This is usually done by selecting the most fit solution (the ones with the best objective function). The algorithm continues operating in the current population and creating new solutions (Childs). Then the new solutions are evaluated to find the best solutions out of them. At the end, the algorithm will be terminated based on a predefined stopping rule and the best found solution will be adopted.

(Leon & Balakrishnan, 1995) described an algorithm to solve RCPSP based on a modified parallel scheduling scheme. The initial solution found by the scheduling scheme was improved by four search procedures (three simple procedures and one genetic algorithm). The GA based search procedures created the new candidate solutions by sampling two solutions from the current population based on their fitness values.

In addition to the *tabu search* and *simulated annealing* algorithms developed by (Lee & Kim, 1996) they have also presented a *genetic algorithm* based on the random key representation scheme and using parallel scheduling scheme as a decoding function to generate the initial set of solutions. The GA used the one-point cross over operator to create the new population of solutions.

(Hartmann S. , 1998) presented three *genetic algorithm* approaches for the RCPSP, based on (priority list, priority rule and the random key representation schemes) using the serial scheduling scheme as a decoding procedure to produce active schedule. The new population was created by using two points crossover.

2.1.2 Robust Resource Constrained Project Scheduling Problem

The previous section has discussed the Resource Constrained Project Scheduling Problem assuming that the activity durations are deterministic and not subject to any disruption or uncertainties. In this section the scheduling under uncertainties will be discussed in addition to the techniques of producing robust schedules secure to little variation in activity durations.

2.1.2.1 Previous literature on RCPSP under uncertainties

The scheduling under uncertainties can be classified under four major approaches, *(Reactive scheduling, stochastic project scheduling, Fuzzy project scheduling and robust (proactive) scheduling*). A survey of the famous literature in those approaches was provided by (Herroelen & Leus, 2005).

In the first approach (*Reactive scheduling*) the uncertainties are not taken into consideration during the generation of the baseline schedule, however this approach suggests revising the baseline schedule when an uncertainty occurs. Various strategies are used in the reactive scheduling approach, starting from simple schedule repair methods such as the right shift rule, to complete or full rescheduling for the uncompleted project activities. (Artigues & Roubellat, 2000) proposed a generic insertion algorithm to add new unexpected activities to the project baseline schedule while keeping the same sequence of previously scheduled activities to minimize the maximum project lateness.

The *stochastic project scheduling approaches* target to create project schedules that minimize the project duration by implementing scheduling policies or strategies but not baseline schedules. Another shortcoming of these approaches is requirement of knowing the probability distribution of the activity durations.

Since the probability distributions for the activity durations are usually unknown, the activity durations in most cases are estimated by human expert judgment. Determining the activities in such way is described by imprecision rather than uncertainty. In these situations, researchers in the field of *fuzzy project scheduling* recommend the use of fuzzy numbers for finding activity durations, instead of stochastic variables. The *fuzzy project scheduling* uses *membership functions* to define the activity durations and to produce the project schedule (Herroelen & Leus, 2005).

With regard to proactive (Robust) scheduling, it accounts for the inconsistency in activity durations caused by uncertainties. Goldratt in 1997 has proposed the critical chain/buffer management (CC/BM) methodology. In this method Goldratt suggested to insert time buffers in the project schedule to protect the project makespan form expanding due uncertainties. His method achieved huge attractiveness to scientists and project managers (Chtourou & Haouari, 2008)

In this thesis more attention will be drawn to the forth type of scheduling under uncertainties (Robust project scheduling). The following section is addressing the literature of this type of scheduling in more details.

Robust (Proactive) Project Scheduling

Abstraction of resource usage

(Herroelen & Leus, 2004) have developed a mathematical model to optimally generate a stable baseline, when a disturbance occurs to one activity only (increase in the duration of one activity). The expected weighted deviation of the sum of starting times of the project activities was used as a stability measure in their model. The authors build their model based on the abstraction of resource use (allocation of resources has been done properly or it will be done in further stage).

In the computational experiments three heuristics were used as a benchmark for the model proposed in the above cited paper. The benchmark heuristics are (*Maximizing the weighted sum of pairwise floats* (WPF), *linear programming based heuristic* (LPH) and the *activity-dependent float factor model*). The findings of the experiments were found satisfactory to the authors since the mathematical model proposed in their paper outperformed the benchmark heuristic in terms of yielding less objective value.

Another paper discussed the stable project scheduling with abstraction of resource constraint is (Van De Vonder, Demeulemeester, Herroelen, & Leus, 2005). The aim of their paper was to discuss the tradeoff between the quality robustness (The schedule ability to absorb uncertainties in project duration measured in terms of project makespan) and the solution robustness (The schedule ability to absorb uncertainties in project duration measured in terms of the variation between the planned activity starting times and the actual starting times in the same project schedule). *Critical Chain Scheduling / Buffer Management* (CC/BM) method was used after modification to create quality robust schedules by concentrating the time buffers at the end of the project *Critical Chain* (CC) (The longest of precedence and resource related activities that determines the project makespan) after reducing the activity duration to its aggressive estimates (based on 50% confidence level). Another type of buffers is added after non-CC activities preceding CC activities to prevent delaying the Critical Chain. The first type of buffers is called *Project Buffer* while the last is called *Feeding Buffer*. The CC/BM method was slightly modified by the authors by crashing the feeding buffers to prevent non-CC activities from starting earlier than the Critical Chain itself.

To create solution robust schedules, the time buffers were distributed through the project schedule, instead of concentrating them to protect the critical chain, and as a result, protecting the project makespan. (Van De Vonder, Demeulemeester, Herroelen, & Leus, 2005) had provided a simulation experiment to recognize when it's recommended to use concentrated buffers (such as in CC/BM and modified CC/BM) and when it is more beneficial to scatter the buffers through the project activities (like in ADFF). The experiments considered multiple factors that may affect the results. Namely the number of project activities ***n***, the order strength ***OS*** and the weighing parameter ***wp*** (defined as the ratio of the end dummy activity weight, to the average weights of all other activities). The authors had fixed a target of 95% TPCP (timely project completion time) and measured the required buffer percentage to achieve this target (for both heuristics ADFF and modified CC/BM). The conclusion was made that ADFF always outperformed the modified CC/BM in solution stability. While in consideration of quality robustness the results have shown that for low values of ***wp***, the modified CC/BM outperforms ADFF, however this advantage for the modified CC/BM over ADFF deteriorate when ***wp*** increases. The modified CC/BM also yielded better results for complex problems (i.e. with larger number of activities ***n*** or larger order strength ***OS***).

Considering the resource constraint

Finding intrinsic robustness

(Chtourou & Haouari, 2008) proposed a two stage priority rule algorithm to create robust schedules with a good makespan. The algorithm aimed to find the best schedule in terms of intrinsic robustness to be capable of absorbing small changes in activity durations.

The algorithm in the first stage solved RCPSP for minimizing the project makespan without considering robustness objective.

Sixteen different priority rules were used for this purpose. The best found solution (after running the algorithm for large number of iterations) was then used as acceptable threshold for the next stage. In the second stage the algorithm was run again for large number of iterations, but this time with the objective of maximizing robustness. For that reason, 12 different robustness measures (RMs) were introduced to measure the schedule robustness. A simulation was conducted to test the robustness of the developed schedules. The simulation found that the schedules with higher RM values outperformed the ones with lower RM values in most of the compared schedules. The paper also introduced an assessment for the performance of robustness measures and priority rules.

Robust Scheduling by adding time buffers

(Van De Vonder, Demeulemeester, Herroelen, & Leus, 2006) had extended their previous study (Van De Vonder, Demeulemeester, Herroelen, & Leus, 2005) to take in consideration the resource constraint. They developed a heuristic procedure for generating stable schedules by inserting time buffers scattered all over the project schedule. The stability measure adopted in their research was minimizing the weighted sum of deviation between the planned starting times of the project activities and the actual starting times found by simulation. The authors have modified the adapted float factor heuristic (ADFF) (used to insert buffers in the non-resource constraint version of the problem) to ensure the resource feasibility of the generated buffered schedule.

The new heuristic starts with a good feasible resource constrained schedule which can be obtained using any exact or heuristic approach. Then to grantee resource feasibility, the resource flow network for the problemat hand shall be found and the network shall be modified by adding extra precedence relationship representing the resource flow. ADFF now can be applied to the new modified network and the resulted schedule will be resource constraint feasible schedule. This new heuristic was called resource flow-dependent float factor (RFDFF).

A computational experiment was conducted to investigate the tradeoff between makespan stability (Quality robustness), obtained by CC/BM heuristic and solution stability (solution robustness), obtained by RFDFF. The experiments take into account the variation in different important parameters, namely the Order Strength (***OS***), the Resource Factor (***RF***) (defined as the average number of resource types used by an activity) and the Resource Constrainedness (***RC***) _ (the average portion of the resource availability that is used by an activity). In addition to weighing parameter ***wp*** and the number of activities ***n***. The simulation method was used to find out what method outperforms the other. Assessed by the percentage of project prolongation time to achieve 95% TPCP, when changing one parameter only of the three main parameters (***OS, RF*** and ***RC***) and fixing the other two parameters.

The authors concluded that the advantages of the two scheduling approaches used in their study to find the trade-off between makespan and stability is highly dependent in the project characteristics, especially the weighing parameter ***wp***. Surprisingly, the research found that the CC/BM method is hard to defend when the project makespan becomes very important, the facts which the authors describe as “paradoxical”. The authors also suggested that even the stable schedules (created using RFDFF) might become infeasible during the project execution and in this case the reactive scheduling methods shall be considered.

Number of heuristic procedures for generating stables project schedules _by including time buffers_ were introduced and validated by (Van de Vonder, Demeulemeester, & Herroelen, 2008). The heuristics studied in their experiment varies between simple heuristics, (specifically The *Virtual Activity Duration Extension* (**VADE**) and *The Starting Time Criticality* (**STC**)). In addition to improvement heuristic (*Steepest Descent*) and *Tabu search*, the resource flow-dependent float factor (**RFDFF**) was used as a benchmark to judge the performance of the tested heuristics. The writers adopted a two stage approach in their study. First, solve the RCPSP problem to find a feasible precedence and resource constraint schedule. Then the heuristic to insert the safety buffers in the second stage was applied to generate stable baseline schedules under the objective of acceptable makespan.

A simulation based analysis was conducted to assess the performance of the proposed heuristics for high, medium and low activity duration variability. The stability cost and the computational time was the bases of judgment on the heuristics performance. The findings of the simulation were in the favor of applying the improvement heuristic starting from the schedule found by ***STC*** heuristic, since it achieved an average stability cost comparable to the ones achieved by the tabu search (found to be the best heuristic in terms of stability cost) but with a very reasonable average computational time comparing to tabu search. The authors have justified these findings by the nature of ***STC*** heuristic which inserts buffers based on the activity weights and duration variability unlike ***RFDFF*** and ***VADE*** which depends on the activity duration variability only. The paper was concluded by recommending robust (proactive) scheduling technique for low activity duration variability problems due to the good results (low stability cost) found in simulation.

Robust scheduling by resources allocation

This method differs from the time buffer method because the stability is achieved by proper resource allocation instead off adding time buffers (Leus & Herroelen, 2004) have developed a branch and bound algorithm to find resource allocation that protects the project makespan from activity duration variability.

2.2 Resource Investment Problem (RIP)

Although RCPSP became a well-known standard project scheduling problem and attracted numerous researchers who developed both exact and heuristic scheduling procedures. however, the RCPSP assumptions are very restrictive for many practical applications. Subsequently, various extensions of the basic RCPSP have been developed. (Hartmann & Briskorn, 2009).

The Resource Investment Problem (RIP), also known as Resource Availability Cost Problem (RACP), is one of the RCPSP extensions and considered as a dual version of RCPSP. The reason behind that; in RCPSP the objective is to minimize the project makespan when the project resources are given. On the other hand, the objective of RIP is to minimize the time-independent availability cost of renewable resources while the project due date is given. In other words, the main goal of RIP is to find a schedule and resource levels such that total resource cost is minimized. (Hartmann & Briskorn, 2009) (Najafi & Azimi, 2009).

2.2.1 RIP Problem Definition

Similar to RCPSP we may define the RIP problem as follows. We assume the graph $\boldsymbol{G}$ represents the activity-on-node (AoN) representation of project scheduling networks, graph $\boldsymbol{G}$ is acyclic. Let $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V} = \boldsymbol{A} \cup \{\mathbf{0}, \boldsymbol{n} + \mathbf{1}\}$ and $\boldsymbol{A} = \{1,\ldots n\}$ is the set of $\boldsymbol{n}$ actual activities (vertices) (activities 0 and n+1 are dummy activities represent the start and end of the project). $\boldsymbol{E}$ is the set of precedence relations between activities (edges). There are $\boldsymbol{m}$ renewable resources $(\boldsymbol{k} = 1\ldots.\boldsymbol{m}) \in \boldsymbol{R}$ (set of renewable resources). The duration of activity $\boldsymbol{i} \in \boldsymbol{A}$ is $\boldsymbol{p_i}$ units of time. During this time period a constant amount of $\boldsymbol{b_{ik}}$ units of resource $\boldsymbol{k}$ is occupied (assigned to activity $\boldsymbol{i}$). The project due date is Δ . Preemption is not allowed and all activities has single execution mode.

The problem is to find a schedule (determine starting or finishing times for the activities) and the required resources $\boldsymbol{B_k}$ (constant from the start to the end of the project) that minimize the total time-independent resource availability cost (***K)*** of the project, while the precedence and resource constraints are satisfied.

The resource investment cost $\boldsymbol{K}$ represented by the multiplication of the number of units required of each type of resource $\boldsymbol{k}$ ($\boldsymbol{B_k}$) by the availability cost per unit of resource $\boldsymbol{k}$ ($\boldsymbol{C_k}$). As per the below equation.

$$\boldsymbol{K = \sum_{k=1}^{m} B_k \, . C_k} \qquad 8$$

In addition to the Resource Cost, Tardiness cost $\boldsymbol{T}$ represents the cost incurred due to late project completion from the predefined due date Δ and may be added to the problem objective. Such cost could represent a penalty, liquidated damages or any other cost which could be incurred due to the project delay.

$$T = max\,(0, \Theta\ - \Delta)\ \pi \qquad 9$$

Where $\boldsymbol{\pi}$ is the amount of cost incurred for each time unit of delay and Θ is the project completion time

Now we can write the total project cost ($\mathbf{Z}$) as follows

$$Z = K + T\ or, \qquad 10$$

$$\boldsymbol{Z = \sum_{k=1}^{m} B_k \ C_k + max\,(0, \Theta\ - \Delta)\pi} \qquad 11$$

2.2.2 Deterministic Resource Investment Problem

As defined earlier the Resource Investment Problem (RIP) is known as the problem dealing with minimizing the resource availability cost of a project within a given deadline. Since RIP belongs to the family of scheduling problem it has proven to be

NP-hard by Möhring (1984). Hence, heuristic algorithms are very commonly applied in solving such problems (Shadrokh & Kianfar, 2007).

(Shadrokh & Kianfar, 2007), developed a genetic algorithm (GA) to solve RIP. Their algorithm permitted the project delay subject to a predefined tardiness penalty. Their algorithm started with randomly generated first generation of solutions. Then the population was divided randomly to two groups and the Childs where created by using three different cross-over operators (one-point, two-points and type 2 two-points). The authors tested their algorithm and compared the results with benchmark instances and reported that the results were satisfactory.

(Ranjbar, Kianfar, & Shadrokh, 2008) have developed two metaheuristics to solve RIP. One was based on Path Relinking algorithm (PR) and the other was a genetic algorithm.

Path relinking algorithm combines two different good feasible solutions (called the initial solution and the guiding solution) together by connecting them based on criteria set in the algorithm.

The research was concluded by a computational experiment that compared the two algorithms to each other and found the results in favor of path relinking algorithm

2.2.3 Resource Renting Problem (RRP)

Resource Renting Problem (RRP) is dealing with the time-dependent resource cost, unlike the Resource Investment Problem (RIP), which deals with the resource cost as time-independent.

The aim of RRP is to find an optimal renting policy for a given schedule that minimize the total resource cost, consists of renting cost and purchase cost (fixed cost applied every time a resource is added or removed).

So the objective of this problem can be written as follows

$$\textbf{Minimize } \boldsymbol{Z} = \sum_{k\in R} \boldsymbol{C_k^p} \sum_{t=0}^{d} \boldsymbol{\alpha_{kt}} + \sum_{k\in R} \boldsymbol{C_k^r} \sum_{t=0}^{d} \boldsymbol{t}(\boldsymbol{w_{kt}} - \boldsymbol{\alpha_{kt}}) \qquad 12$$

Where α_{kt} and w_{kt} are the number of units added or removed from resource ***k*** respectively. C_k^r is the renting cost of resource ***k*** per unit of time, and C_k^p is the per unit procurement cost of resource ***k.***

(Nubel, 2001) has introduced a branch and bound algorithm that enumerates a finite number of schedules _proved to contain an optimal one_ with given deadline, to find the optimal renting policy which yields the lowest total resource cost, composed of both (purchase time independent cost and renting time-dependent cost). A comprehensive experiment was done on the algorithm and the author reported that the solution concept is promising.

2.2.4 Robust Resource Investment Problem

The Literature in robust solutions for RIP is limited and has huge room for future research. (Yamashita, Armentano, & Laguna, 2007) have discussed the difference between solution-robust solutions (Which is defined as the solution which remains optimal or close to optimal for any problem scenario), and model-robust solutions (defined as the solution which remains feasible or almost feasible for any scenario).

The paper proposed a solution-robust algorithm to solve (RACP) which is robust to all scenarios represented by sets of different activities duration.

The algorithm was developed based on scatter search procedure and employed strategies like path relinking and dynamic updating in reference set. The algorithm was applied to two different models with different robustness objective functions (minimize of the maximum regret and minimization the mean-variance).

The results of the computational experiments showed that the models produced conflicting solutions.

3 Methodology, Finding A Robust Solution for Resource Investment Problem with Time-Dependent Resource Cost and Tardiness Penalty (RIP-TDRC)

In this chapter the methodology for finding a robust solution for the Resource Investment Problem with Time-Dependent Resource Cost and Tardiness Penalty (RIP-TDRC) will be discussed in details.

As discussed earlier in chapter 1. The difference between RIP-TDRC and RIP is that the former deals with time-dependent resource cost but the latter deals with time-independent resource cost.

In this chapter, first the deterministic version RIP-TDRC will be discussed. Then the deterministic solution will be modified and used by inserting it into simulation algorithm to find a robust solution for the same problem.

3.1 Solving the deterministic RIP

The Problem Formulation

RCPSP formulation of (Pritsker, Watters, & P.Wolfe, 1969) is used as the base for our problem formulation. The objective function was changed to suite RIP-TDRC and additional constraints where added to make the formulation valid.

Before starting with the formulation few initial setups shall be done as they deemed to be necessary to apply the above mentioned mathematical formulation.

1. Θ_{min} , the minimum project completion time for the problem on hand and the early start times for each activity ES_i shall be calculated. The same can be

found by neglecting the resource constraint and solve the problem as ordinary PSP using CPM calculation.

2. Θ_{max}, the maximum completion time of the project. Shall be also calculated. This can be accomplished by solving the corresponding RCPSP using (Model 1) after setting $\boldsymbol{B_k} = max_{\ i \in A}\{\ b_{ik}\}$ for each $\boldsymbol{k} \in \boldsymbol{R}$

3. Similar to $\mathrm{ES_i}$ the value of $\mathrm{LF_i}$ and $\mathrm{LS_i}$ can be found by solving the backward pass of the problem neglecting the resource constraint using CPM method starting from Θ_{max}.

3.1.1 The Mathematical Model

Model 2: RIP-TDRC Mathematical Model

The objective function

$$\textbf{Minimize } \boldsymbol{Z} = \boldsymbol{K} + \boldsymbol{T} = \sum_{\boldsymbol{k}=\boldsymbol{1}}^{\boldsymbol{m}} \sum_{\boldsymbol{t}=\boldsymbol{1}}^{\Theta_{\boldsymbol{max}}} \boldsymbol{t}.\hat{\mathrm{C}}_{\boldsymbol{k}}.\boldsymbol{B_k}\,\boldsymbol{x_{n+1,t}} + \sum_{\boldsymbol{t}=\boldsymbol{1}}^{\Theta_{\boldsymbol{max}}} \boldsymbol{\pi}.\,\mathbf{max}\,(\boldsymbol{0}, \boldsymbol{t}.\boldsymbol{x_{n+1,t}} - \Delta) \qquad 13$$

Where,

$\boldsymbol{x_{i,t}}$, is a binary decision variable equals to 1 if activity ***i*** starts at time unit ***t***. And $\hat{\mathrm{C}}_{\boldsymbol{k}}$ is per unit cost of resource ***k*** for each unit of time.

Subject to the following constraints

$$\sum_{t=ES_j}^{LS_j} \boldsymbol{t.x_{jt}} \geq \sum_{t=ES_i}^{LS_i} \boldsymbol{t.x_{it}} + \boldsymbol{p_i} \qquad \forall\, (\boldsymbol{i},\boldsymbol{j}) \in \boldsymbol{E} \qquad 14$$

$$\sum_{i=1}^{n} \sum_{\tau=\max(ES_i,t-p_i+1)}^{\min(LF_i,t)} \boldsymbol{b_{ik}\, .x_{i\tau}} \leq \boldsymbol{B_k} \qquad \forall\, \boldsymbol{t} \in \boldsymbol{H},\quad \forall\, \boldsymbol{k} \in \boldsymbol{R} \qquad 15$$

16

Where, H is the scheduling horizon starts a t =0 to θ_{max} and R is the set of renewable resources.

$$\sum_{t=ES_i}^{LS_i} \boldsymbol{x_{it}} = \mathbf{1} \qquad \forall\, \boldsymbol{i} \in \boldsymbol{A} \cup \{\boldsymbol{n}+\mathbf{1}\} \qquad \mathbf{16}$$

$$\boldsymbol{x_{00}} = \mathbf{1} \qquad 17$$

$$\boldsymbol{x_{it}} = \mathbf{0} \qquad \forall\, \boldsymbol{i} \in \boldsymbol{A} \cup \{\boldsymbol{n}+\mathbf{1}\},\ \ \boldsymbol{t} \in \boldsymbol{H} \backslash \{\boldsymbol{ES_i}, \ldots., \boldsymbol{LF_i}\} \qquad 17$$

$$\boldsymbol{x_{it}} \in \{\mathbf{0}, \mathbf{1}\} \qquad \forall\, \boldsymbol{i} \in \boldsymbol{A} \cup \{\boldsymbol{n}+\mathbf{1}\},\ \ \boldsymbol{t} \in \{\boldsymbol{ES_i}, \ldots., \boldsymbol{LF_i}\} \qquad 18$$

Constraints (14) and (15) are the precedence constraints and resource constraints respectively.

Constraints (16)and (19)impose non-preemption of the project activities. Since ES_i stands for the earliest start time for activity $\boldsymbol{i}$, and $\boldsymbol{LS_i}$ is the latest possible start time activity $\boldsymbol{i}$.

Constraints (18) mean that the start time of any activity can only occur between its earliest start time and its latest start time.

The alert reader can easily note that the objective function is not linear since both B_k and $\boldsymbol{x_{n+1,t}}$ are decision variables. Because in RIP-TDRC the cost of the resource is time-dependent so this resulted to non-linear objective function.

To avoid this problem, the well-known large integer trick will be used by replacing the two decision variable by one new variable

$$\boldsymbol{y_{kt}} = \boldsymbol{x_{n+1,t}} \,.\, \boldsymbol{B_k} \qquad 19$$

Now we may rewrite the objective function to be

$$\textbf{Minimize}, \boldsymbol{Z = K + T} = \sum_{\boldsymbol{k=1}}^{\boldsymbol{m}} \sum_{\boldsymbol{t=1}}^{\theta_{\boldsymbol{max}}} \boldsymbol{t}.\hat{\text{C}}_{\boldsymbol{k}}.\boldsymbol{y_{kt}} + \sum_{\boldsymbol{t=1}}^{\theta_{\boldsymbol{max}}} \boldsymbol{\pi}.\boldsymbol{max}\,(\boldsymbol{0}, \boldsymbol{t}.\boldsymbol{x_{n+1,t}} - \Delta) \qquad 20$$

A set of new constraints shall be added to the original set of constraints (14) to (19) to force the value of y_{kt} to be either equals to 0 or B_k.

$$y_{kt} \leq u_k \,.\, x_{n+1,t} \qquad \forall\, t \in H, \quad \forall\, k \in R \qquad 22$$

$$y_{kt} \leq B_k \qquad \forall\, t \in H, \quad \forall\, k \in R \qquad 23$$

$$y_{kt} \geq B_k - u_k\,(1 - x_{n+1,t}) \qquad \forall\, t \in H, \quad \forall\, k \in R \qquad 24$$

$$y_{kt} \geq 0 \qquad \forall\, t \in H, \quad \forall \in R \qquad 25$$

Where $\boldsymbol{u_k}$ is a large integer and shall be selected to be larger than any possible value of B_k. We note that constraint (22) and (25) will set $\boldsymbol{y_{kt} = 0}$, when $\boldsymbol{x_{n+1,t}}$=$\mathbf{0}$. While when $\boldsymbol{x_{n+1,t}}$=**1** constraints (23) and (24) will force to set $\boldsymbol{y_{kt} = B_k}$.

Selecting the value of $\boldsymbol{u_k}$

It is very important to select the values of the new large integers u_k carefully. It is clear from constraints (22) to (24) that selecting a low value of u_k could lead to infeasible solution (if $u_k < max_{\,i \in A}\{ b_{ik}\}$) or could lead to derive the solution by bounding the value of $\boldsymbol{B_k}$.

On the other hand, selecting a very large value of u_k will lead to long computation time. So for that purpose, we set u_k to be equal to the sum of all resource requirement of type $\boldsymbol{k}$ for all activities**.** This assumes the worst case of having all activities scheduled together. So we set;

$$u_k = \sum_{i=1}^{n} b_{ik} \qquad 26$$

Also note that u_k act as an upper bound for the decision variables $\boldsymbol{B_k}$ as well.

Upper and Lower bound for the decision variables

After finding a suitable upper bound for $\boldsymbol{B_k}$ we need to find a suitable lower bound for it, and since $\boldsymbol{B_k}$ cannot be less than $max_{\,i \in A}\{ b_{ik}\}$ we can add the following lower bound to our formulation.

$$B_k \geq max_{\,i \in A}\{ b_{ik}\} \qquad 27$$

It is worth mentioning that the other decision variables x_{it} are already bounded by the early starting time and early finish time of each activity as in constraints (18).

Note: The above problem formulation contains **(n . Θ_{max}) + (m . Θ_{max}) + m** decision variables.

3.2 Solving the stochastic RIP version to find a robust solution

In the previous section we have assumed that the activity durations are fixed and not subject to change due to any reason. Unlike the situation in actual life where the uncertainties exist very often.

The concept of simulation optimization was used in this section to find a solution for the stochastic version of RIP-TDRC which is robust and immune against slight deviation in activities duration.

An algorithm was written to find a robust solution by finding a schedule and resource assignment represented by the project makespan $\hat{\Theta}_{opt}$ which is expected to yield the least cost, out of all possible Makespans of the deterministic RIP-TDRC version (i.e. from Θ_{min} to Θ_{max})

To explain the idea in more details, let us discuss a small project (Example 1) shown in Figure 1.

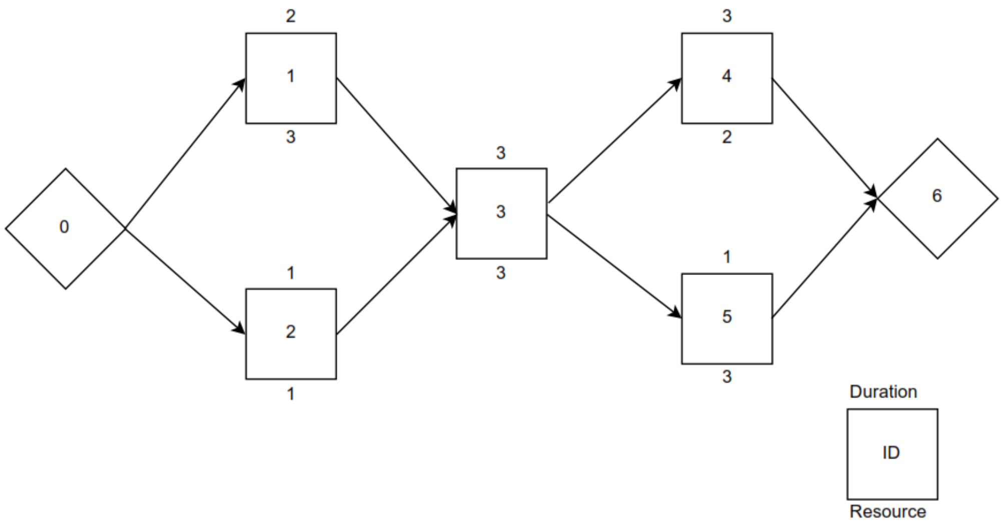

Figure 1: Example 1

The project has 5 non-dummy activities. For simplicity we assume that we have one type of renewable resources only $\boldsymbol{r_1}$ **,** and has the cost of 1 per unit of time for each unit of resource $\hat{c}_{\mathbf{1}} = \mathbf{1}$.

The project has preset deadline of (Δ=10). A penalty of ($\pi = 10$) will be applied for each time unit of delay after the project deadline.

Note that the project has minimum completion time of $\Theta_{min} = 8$, found by solving the problem using CPM method (i.e. having infinite resources). The maximum completion time of the project is $\Theta_{max} = 10$, achieved if we have the minimum resources requirement of B_1= 3.

Solving the problem assuming fixed activity durations, will yield the optimal solution shown in Figure 2 with objective function $Z = 30$, $\Theta_{opt} = 10$, B_1= 3. Note that this example has more than one optimal solution.

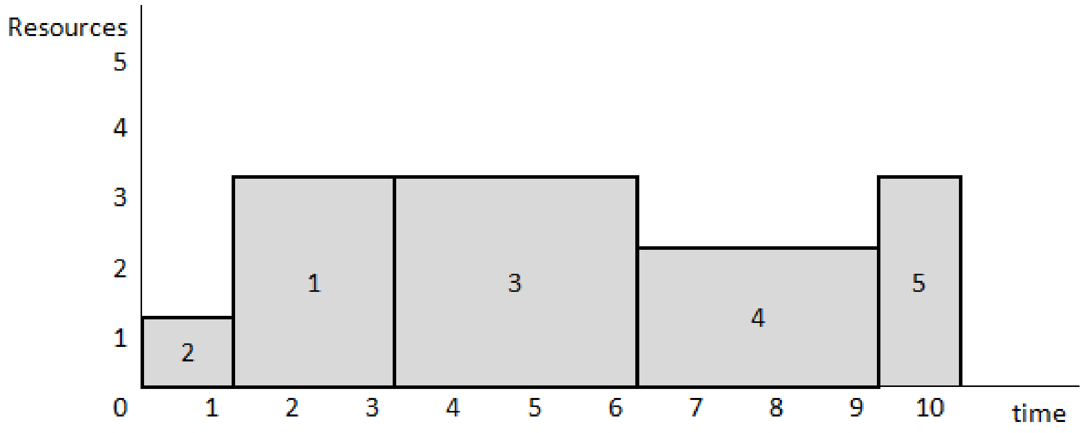

Figure 2: Optimal Solution for example 1

Now, if we decided to proceed with this schedule as project base line, and we have encountered an unexpected delay in activity 2 for one-time unit only. Then the entire project duration will be delayed by one-time unit also and the project cost Z will increase to **Z**=43. Where ***K***= 11 x 3 x 1=33 and ***T***=10. See Figure 3.

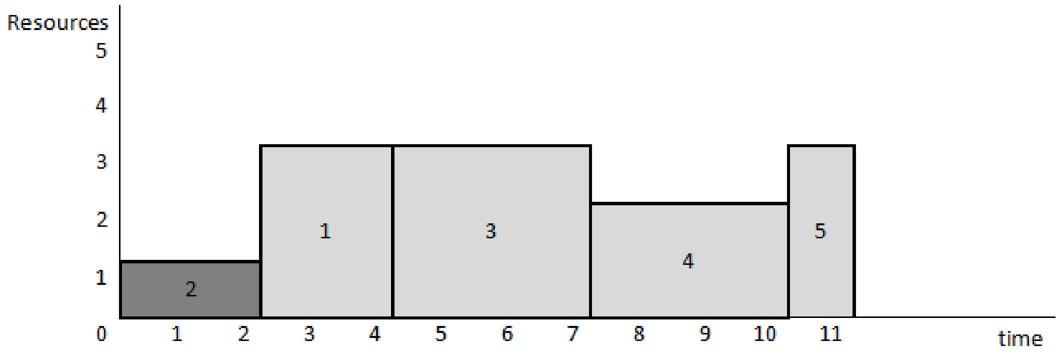

Figure 3: Example 1 solution when $\Theta_{\boldsymbol{opt}} = \mathbf{10}$ after the increase in duration of activity 2

Note that any other increase in activity duration (Either for the same activity or any other activity) will cause the total project cost to be **Z**=56.

Similarly, any activity duration increase will increase the project cost by 13 units.

To test the solution Robustness let us consider the case of planning the project based on selecting the project duration to be 9 instead of 10. This can be achieved by solving the problem using Model 2 with adding the constraint in Equation (28). The solution found (shown in Figure 4) has an objective function of $\boldsymbol{Z} = 36$ and $\boldsymbol{B_1} = 4$.

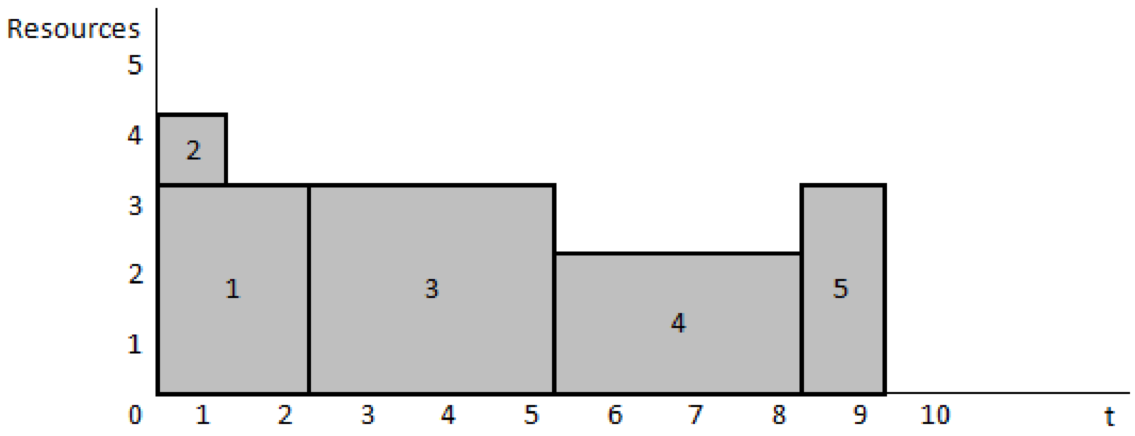

Figure 4: Example 1 solution at $\Theta_{\boldsymbol{search}} = \mathbf{9}$

Note that here in this case, an increase in activity 2 duration by one-time unit will not cause any increase in the project duration. So, the realized objective function will remain $\boldsymbol{Z} = 36$.

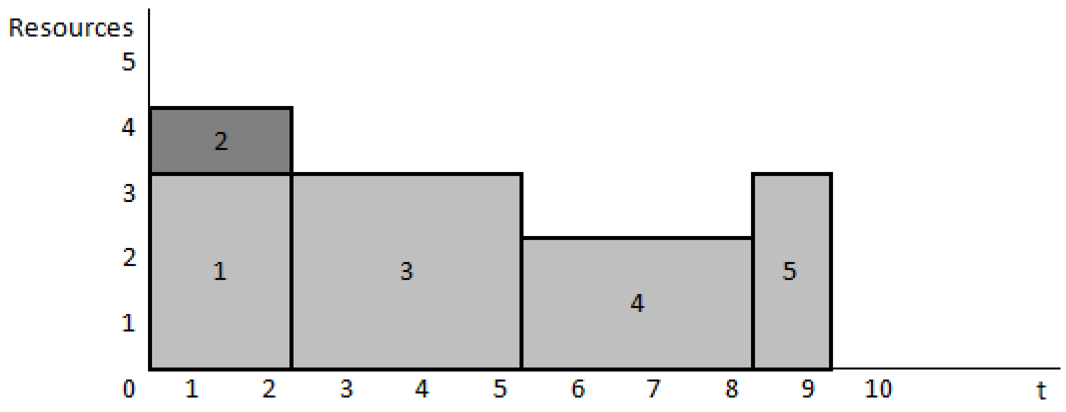

Figure 5: Example 1 solution at $\Theta_{search} = 9$ after the increase in duration of activity 2

Note that selecting the project completion date to be $\Theta_{search} = \mathbf{9}$ not only yielded more robust solution but also a better project cost.

From here we can understand the importance of solution robustness. In our example (case 2) the project manager do not need to take any action to recover the delay in activity 2. No additional resources to be added and all other activities started as planned, not like case 1, where all the other activities were rescheduled, the project was delayed and the cost has increased.

This is just an example for illustration, in the actual life we cannot know exactly what could happen and what activity will be delayed. So, in our proposed solution simulation will be used and conducted for large number of iteration before selecting the optimal schedule and assignment represented by the makespan $\hat{\Theta}_{opt}$.

Description of the General Algorithm used to find the robust solution is as follows:

1. Read the all project data and parameters.

2. Find Θ_{min} and ES_i by solving the problem using CPM method (Without considering the resources).
3. Find Θ_{max} by solving RCPSP and setting the resources to the minimum required to complete the project ($\boldsymbol{B_k} = max_{\ i \in A}\{\ b_{ik}\}$ for each $\boldsymbol{k} \in \boldsymbol{R}$.).
4. Find latest finish times LF_i by solving the backward pass of the problem from Θ_{max}.
5. Find the solution for the deterministic version of RIP-TDRC using Model 2: RIP-TDRC Mathematical Model
6. Find the best observed solution for the non-deterministic version or RIP-TDRC by exploring the solution space from Θ_{min} to Θ_{max} as follows:

   6.1 Solve RIP-TDRC exactly by imposing Θ_{search} (Starting from Θ_{min}) as completion time. By adding the following constraint to Model 2.

   $$\sum_{t=\boldsymbol{ES_{n+1}}}^{LF_{n+1}} \boldsymbol{t}.\boldsymbol{x_{n+1,t}} = \quad \Theta_{search} \qquad 28$$

   6.2 Find the realized starting times using Monte Carlo simulation (The simulation algorithm is described below) by allowing a limited random increase in the durations of a percentage of randomly selected activities.

   6.3 Repeat from step 6.1 above with replacing $\Theta_{search}\ by\ \Theta_{search} + inc$ until the whole solution space is explored and $\hat{\Theta}_{opt}$is found.

Note that the parameter ***inc*** represents the increase in the makespan value for each iteration during the search procedure to find the best solution (makespan) from Θ_{min} to Θ_{max}. The value of ***inc*** can be selected depending on the problem structure, number of activities, the possible makespan range..etc.

In our computational experiment we have selected ***inc*** to be one-time unit to be able to explore all the possible makespans. This is feasible in our experiment since we have conducted the experiments on instances with 20 activities (Refer to chapter 4), so the gap between Θ_{min} and Θ_{max} is relatively small. However, in large problems the gap between makespans could reach to 100s of even 1000s of time units so it is very practical on such cases to set ***inc*** to a larger value.

3.2.1 The Simulation Process

As explained in general algorithm points 6.1 to 6.3. The simulation will take place after finding the exact solution for a given project finish time $\Theta_{\boldsymbol{search}}$.

The exact solution will provide the simulation algorithm with activity starting times (x_{it}) and available resources $\boldsymbol{B_k}$. Other parameters like original activity durations, precedence relationships and resource requirements for each activity will be also entered to the simulation algorithm.

The simulation algorithm will use Parallel Scheduling Scheme (SGS) with activity earliest starting time as priority rule and activity ID as tie break rule to reschedule the activities as explained below.

Start of Simulation algorithm

For each iteration from 1 to itr, {

- The algorithm based on preset simulation parameters will find the new activity durations (for randomly selected activities to be delayed), those parameters will be discussed in details in chapter 4.

- The algorithm will start progressing on time from time 0 (The time counter will stop after all activities are being scheduled).

  For each time iteration from t=0 until t= starting time of activity n+1 {

    - All activities will be checked if eligible for scheduling or not based on the following conditions:
        1. Is the Starting time provided from the exact solution due?
        2. Is the precedence constraint respected (are all the precedence activities completed)?
    - The activities meeting the above conditions enter the eligible set and will be candidate for scheduling.
    - The algorithm will select an activity for scheduling from the eligible set based on the earliest planned starting time (from the exact solution) as priority rule. The resource constraint will be checked before scheduling. And the resource availability will be updated after selecting the activity for scheduling.
    - Then the activity with next priority in the list will be checked against resource availability to be scheduled if we still have enough resources. This step will be repeated until no activity can be scheduled in the eligible set.
    - Now the timer can be increased by one unit and for every time increment the algorithm checks for completed activities and update the resources.
    - This process will be repeated until all activities are scheduled.

    }

- Now we have one realized schedule, the completion time will be saved and the objective function will be computed and saved.
- The iteration counter now can be increased by 1 and the process will be repeated for the set number of iterations (itr).

  }

After all iterations are completed the average realized project completion time and the average objective function for all the iterations will be computed.

End of Simulation Algorithm.

The whole procedure will be repeated for all possible makespans found by the exact solution ($\Theta_{min} \leq \Theta_{search} \leq \Theta_{max}$).

The solution (i.e. schedule and resource assignment) which has less realized cost, will be the recommended solution as proven by the simulation.

Figure 6 below illustrate the idea more clearly. The figure represents the solution for one of the instances solved during the computational experiment (J208-2-E). The blue function represents the objective value (cost) for the deterministic problem version found by MILP model for all possible makespans in this instance (From 31 to 53). Whereas the red function represents the cost found using the simulation algorithm considering the uncertainties.

It is noticeable that the optimal solution (schedule) found by MILP model (1974 at makespan 43) is not the optimal solution after considering the uncertainties, since this schedule resulted on a cost of 3078 if selected as baseline, while the proposed algorithm recommended schedule found to be with a makespan of 38 and expected to yield a cost of 2335, 24.1% better than the MILP solution.

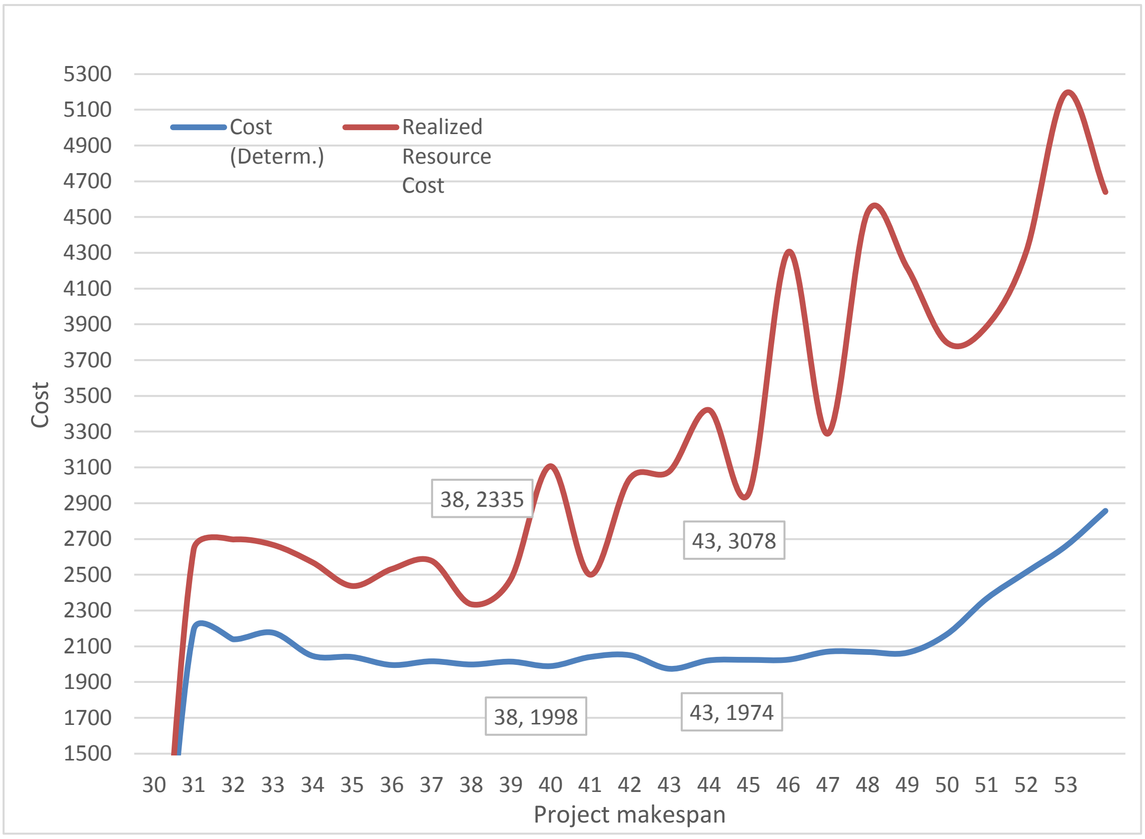

Figure 6: Solution of example 2 (instance 208-2-E)

Next chapter will discuss the computational experiments applied the algorithm explained in this chapter.

4 Implementation (Computational Experiment)

4.1 Selecting the parameters

Before proceeding with the experiment it is a must to decide some important parameters values that may affect the results of solving the models. The parameters of concerns are:

1. **Project deadline** Δ : represented by time units, if the project makespan exceeds this deadline a tardiness penalty shall be applied.
2. **Cost per unit of resource per unit of time** $\hat{C}_{\boldsymbol{k}}$**.**
3. **Tardiness penalty for each unit of time delay** $\boldsymbol{\pi}$: For each unit of time the project is delayed from the preset deadline Δ, an amount of tardiness penalty $\boldsymbol{\pi}$ will be applied.
4. **Probability of activity being delayed by simulation algorithm** $\boldsymbol{D_n}$.
5. **Allowable delay in activity duration** $\boldsymbol{T_m}$

Selecting a suitable Δ for our problem is very important since the value of Δ is a deriving factor in the problem objective function. As explained earlier the project makespan in the deterministic problem can be any value between Θ_{min} and Θ_{max} however in the non-deterministic version of the problem the makespan may exceeds Θ_{max} due to the increase in some activities durations.

For the above reason two values for Δ were selected for the experiment, one value near Θ_{min} (tightly constrained) and the other near Θ_{max} (loosely constrained) as follows:

$\Delta_1 = \Theta_{min} + ((\Theta_{max} - \Theta_{min}) \text{ x } 0.3)$

and

$\Delta_2 = \Theta_{min} + ((\Theta_{max} - \Theta_{min}) \text{ x } 0.8)$

The problem will be solved for the selected values of Δ and fixing the other parameters, to evaluate the effect of changing the project deadline on the problem solution.

With regard to Cost per unit of resource per unit of time $\hat{C}_{\boldsymbol{k}}$, each instance will be solved four times with four different sets of resource cost . First the resource cost per unit of time for each type of resource $\boldsymbol{k}$ will set to be equal$\hat{C}_{\mathbf{1}} = \hat{C}_{\boldsymbol{k}} = \cdots \hat{C}_{\boldsymbol{m}}$. For the other three sets of $\boldsymbol{C}$ values $\hat{C}_{\boldsymbol{kmax}} / \hat{C}_{\boldsymbol{kmin}}$ were selected to be equal to 3, 5 and 10 respectively.

The sets of resource cost are selected to be as follows

$\boldsymbol{C_1} = \{ 1, 1, 1, 1\}$

$\boldsymbol{C_2} = \{ 1, 2, 2, 3\}$

$\boldsymbol{C_3} = \{ 1, 3, 4, 5\}$

$\boldsymbol{C_4} = \{ 1, 4, 7, 10\}$

To calculate **Tardiness penalty for each unit of time delay $\boldsymbol{\pi}$** we try to find a reasonable value which is in line with the standard practice in project management field. In general practice the amount of tardiness penalty is usually limited to a 10% of the project value (usual practice in construction industry). However, in our experiment there will be no limit for tardiness penalty. But the 10% penalty rule will be used to be the amount of total penalty if the project duration reached to Θ_{max}.

Since our problem considers only the cost of renewable resources and based on the assumption that the renewable resources represent around 25% of the total project value (this assumption is also based on construction industry), then the value of the tardiness penalty can be estimated to be limited to 40% of the renewable resource cost. Calculated as follows

Total penalty to project cost = 10%

Total resource cost to project cost =25%

So, Total penalty to resource cost = penalty to project cost / resource cost to project cost

= 10% / 25% = 40%

Now to find $\boldsymbol{\pi}$**,** we need first to decide in how many units of time the maximum penalty will be reached. In other words, after how many units of time the penalty will reach the 10% of the project value or 40% of renewable resource value?

As discussed above Θ_{max} is the selected point to reach to 40% penalty out of renewable resource value

Based on the above the value of $\boldsymbol{\pi}$ can be found as follows:

1. The value of the project resource cost $\boldsymbol{K}$ will be calculated first for the deterministic version of the problem.
2. Then the value of $\boldsymbol{\pi}$ will set to

$$\boldsymbol{\pi} = (40\% * \boldsymbol{K}) / (|\Theta_{max} - \Delta|)$$

Note that the value of π is not independent from the project deadline

The Probability of an activity being delayed $\boldsymbol{D_n}$ and Allowable delay in activity duration $\boldsymbol{T_m}$ are set to be 25% and (10% to 25%) respectively. Last the number of iteration for each simulation loop was selected to be itr =5000.

After setting all the parameters the computational experiments may start. For that reason, IBM-Compatible PC was used with Windows 10 installed as operating system. The PC CPU is (64 bit, Intel Core i7-4720HQ CPU @2.6 GHz) and equipped with 16 GB Random Access Memory.

The algorithms were coded using Java 8.9 Programming language with NetBeans IDE 8.0.2 editor. IBM ILOG CPLEX Optimization Studio 12.6.2 solver (run on Java platform) was used to solve the MILP models.

4.2 The Problem Instances

Since the problem at hand can be considered as relatively new problem, benchmark instances with known solutions were not found in literature. For that reason, selected RCPSP instance form PSPLIB (Demeulemeester & Herroelen) were used in the experiment.

At the beginning selected instances with 30 non-dummy activities (J30 set) were tried for the experiment. Unfortunately, due to the problem complexity the algorithm did not converge to optimal solution or near optimal solution in reasonable time for most of the instances.

This leads us to try reducing the (J30 Set) instances from 30 to 20 non-dummy activities by deleting the last 10 activates (21 to 30) and connecting the activities left without successor to the project end dummy activity. The instances were renumbered due to the modification to indicate the new number of non-dummy activities by replacing the number 30 in the original instance number by 20. For example:

Instance original number: J301-1

The New number is: 201-1

All the instances used in the experiments can be found in Appendix 1.

The tests were then conducted on 70 of the modified instances. But the results of 44 of them were disqualified due very long run time or due to the small difference between minimum and maximum makespans for those instances.

For the qualified 26 instances, the experiment was conducted for eight times each, with different parameters (4 different cost assignment and two different penalty value) as discussed earlier in this chapter. Each one of the eight sub-instances was suffixed with a letter from A to H, the attributes of the sub-instances are summarized in

Table 2 below

Table 2: Attributes of the experiment sub-instances

Instances suffix	**Project Deadline**	**Resource Cost**
A		[1 , 1 , 1 , 1]
B	min + (max-min) x 0.3	[1 , 2 , 2 , 3]
C	(Tightly Constrained)	[1 , 3 , 4 , 5]
D		[1 , 4 , 7 , 10]
E		[1 , 1 , 1 , 1]
F	min + (max-min) x 0.8	[1 , 2 , 2 , 3]
G	(Loosely Constrained)	[1 , 3 , 4 , 5]
H		[1 , 4 , 7 , 10]

All instances were solved in reasonable run time. (187.8 Sec in Average) refer to Table 3 below summarizing the run time for each instance

In the next Chapter the findings of the computational experiments will be discussed in details.

Table 3: Run time for each problem Sub-instance

Instance	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**
202-2	103.0	62.7	52.3	45.1	47.2	47.1	42.7	51.1
203-2	61.7	37.7	54.3	44.2	42.8	56.7	45.6	43.3
205-1	113.0	78.4	308.1	1160.5	78.4	96.9	205.6	87.7
205-2	267.4	268.7	288.9	247.8	251.6	321.2	304.6	520.4
206-1	376.8	316.2	242.7	189.6	263.5	299.5	325.9	213.5
206-2	397.7	362.2	357.2	338.7	412.3	385.6	330.0	486.3
207-1	88.6	80.4	66.8	54.8	63.6	60.2	73.1	37.8
207-2	1275.2	1067.8	960.6	1011.7	1255.3	1159.0	1161.0	1117.0
208-1	196.4	129.7	126.1	139.9	192.9	156.8	124.9	173.3
208-2	512.9	422.3	325.1	269.6	614.1	365.4	407.7	264.4
2017-1	99.4	107.4	116.2	92.3	88.9	95.8	90.1	132.7
2018-1	84.8	79.3	84.1	49.9	44.8	49.8	48.3	47.2
2019-1	64.9	70.3	48.9	108.6	42.8	158.5	41.7	66.5
2019-2	71.6	103.9	49.2	51.2	38.3	44.6	63.9	44.2
2020-1	49.2	50.1	121.1	70.9	74.9	69.3	28.2	138.9
2022-1	140.6	166.2	128.8	150.3	169.9	162.7	149.1	152.2
2023-1	106.5	102.2	104.5	106.0	108.4	105.1	120.2	129.4
2023-2	125.1	99.8	96.0	99.0	86.9	81.7	110.3	100.6
2024-1	125.7	124.3	143.1	138.9	144.9	127.9	129.4	102.4
2024-2	218.9	168.4	155.4	149.5	236.6	157.1	157.3	171.6
2033-1	83.4	94.7	81.3	68.6	72.8	90.2	76.8	110.6
2033-2	135.6	32.6	31.8	36.8	126.5	30.5	17.2	25.4
2035-2	56.9	37.9	41.3	39.9	41.1	85.9	46.6	43.0
2036-1	88.8	79.1	76.5	83.0	79.0	92.2	67.5	66.8
2036-2	71.7	65.3	59.6	68.2	78.9	65.8	88.4	115.7
2037-1	400.5	432.4	391.5	373.6	424.3	424.3	432.0	412.7

5 Discussion

In this chapter the findings of the experiments will be discussed in detail. It is worth to mention here that the reasons for conducting the computational experiment were mainly to validate the model and to study the effects of changing major parameters, such as resource cost, penalty value and project deadline. The solution robustness will be also discussed in this chapter as well.

Unfortunately, due to the use of modified instance, the effect of other network and resource related parameters such as Order Strength (OS), Resource Factor (RF) and Resource Constrainedness (RC), cannot be identified.

5.1 The Deterministic Problem Vs the Stochastic Problem Optimal Solution

Before discussing the factors affecting the solution, let us first discuss the solutions provided by the proposed algorithm and the differences between the solutions found in both problem versions, the deterministic and the stochastic.

The solution will be discussed from two aspects only, the project makespan and the project cost (objective function). Other aspects like activity starting times are beyond the scope of this discussion, although it is considered as an important factor, especially when considering solution robustness.

As a start it is very important to clarify the difference between the recommended completion time $\hat{\Theta}_{opt}$ and realized completion time $\Theta_{\boldsymbol{act}}$, since these two abbreviations will be repeated frequently in the discussion. The first one $\hat{\Theta}_{opt}$, is the project recommended completion time related to the schedule and the resource assignment that minimize the realized objective function $\boldsymbol{Z}$ after simulation exploring all possible makespans, while the latter is the realized completion time after running the simulation. In other words $\hat{\Theta}_{opt}$ is the project completion time which if imposed in the algorithm it will result in the least objective function. But $\Theta_{\boldsymbol{act}}$ is the realized project completion resulted from simulation for the same imposed project completion time $\hat{\Theta}_{opt}$.

Similarly, we have to introduce the cost associated with each of the above mentioned project completion times.

1. $\boldsymbol{Z_{opt}}$ **:** The optimal cost found by the MIP model for the deterministic problem.
2. $\boldsymbol{Z_{act}}$ Realized Project Cost found by Simulation if the project was scheduled based on the optimal solution related to $\Theta_{\boldsymbol{opt}}$
3. $\acute{Z}_{\boldsymbol{opt}}$ The realized project cost found by the simulation algorithm for the stochastic problem version, it corresponds to the schedule found by imposing completion time to $\hat{\Theta}_{opt}$

With regard to the objective function, the experiments show increase in objective function values $\acute{Z}_{opt}$ and $\boldsymbol{Z_{act}}$ in the stochastic problem version compared to the deterministic one. This is an expected result and justified by the increase in project duration due to the increase of activity durations, which results in increase in resource cost $\boldsymbol{K}$. Also note that the increase in the project duration will increase the probability of entering the tardiness zone or increasing the tardiness cost $\boldsymbol{T}$.

Table 4Summaries the objective values in both problem variants and the average increase in the objective value of the stochastic variant $\acute{Z}_{opt}$in comparison with the deterministic variant $\boldsymbol{Z_{opt}}$ for all sub-instances.

Table 4: Experiment Objective Values Summary

	Average $\boldsymbol{Z_{opt}}$	**Average $\boldsymbol{Z_{act}}$**	**Average $\acute{Z}_{opt}$**	**Av. $\boldsymbol{Z_{act}}$ – Av. $\acute{Z}_{opt}$**	**% Inc. $\boldsymbol{Z_{act}/Z_{opt}}$**	**% Inc. $\acute{Z}_{opt}/\boldsymbol{Z_{opt}}$**
A	1698.2	2464.0	2358.5	105.46	45.10%	38.89%
B	3375.8	4841.7	4679.5	162.23	43.42%	38.62%
C	5433.2	7768.9	7558.0	249.27	42.99%	39.11%
D	9103.4	13142.1	12853.7	288.38	44.36%	41.20%
E	1679.0	2767.9	2323.7	444.23	64.85%	38.39%
F	3286.4	5127.3	4556.7	570.58	56.02%	38.65%
G	5381.8	8349.9	7372.3	977.62	55.15%	36.99%
H	8947.0	14877.4	12477.8	2399.54	66.28%	39.46%

Note that the increase in the objective value $\acute{Z}_{opt}$ in the stochastic version was not affected much by the sub-instances parameters. The increase was between 36.99% and 41.20%. This little variation in the deviation between $\boldsymbol{Z_{opt}}$ and $\acute{Z}_{opt}$ is caused by the fixed simulation parameters for all sub-instances.

The increase in objective function value $\acute{Z}_{opt}$ in the stochastic problem version justifies the next observation, which is the project recommended completion time $\hat{\Theta}_{opt}$ found by simulation tends to be less than $\Theta_{\boldsymbol{opt}}$ found by the exact method.

The observation is justified by the model tendency to plan the project to finish earlier by adding more resources to be more protected against extra cost resulted from tardiness penalty, or even extra resource cost due the increase in the realized project completion Θ_{act} which is always greater than $\hat{\Theta}_{opt}$.

In fact this technique is equivalent to the well-known time buffer technique used to increase robustness in RCPSP but in our problem RIP-TDRC increasing resources will protect the project from the increase in project cost resulted from the increase in makespan.

In practice that is usually the case, project managers tend to add more resource when the project activities became subject to high uncertainties.

below illustrates the above observation.

Table 5: Experiment Project Completion Time Θ_{opt} and $\hat{\Theta}_{opt}$

	Average Θ_{opt}	**Average $\hat{\Theta}_{opt}$**	**% Decrease**
A	35.35	34.15	3.49%
B	35.19	34.73	1.33%
C	35.15	34.69	1.33%
D	35.27	34.58	2.00%
E	37.27	35.42	5.21%
F	37.31	35.50	5.09%
G	37.54	35.65	5.29%
H	37.50	35.69	5.06%

Please note that the decrease in $\hat{\Theta}_{opt}$ might become more significant if all the optimal solutions for the deterministic problem Θ_{opt} were greater than Θ_{min} because in the instances where $\Theta_{opt} = \Theta_{min}$ the value of $\hat{\Theta}_{opt}$ cannot be decreased any further.

5.2 Solution Robustness

Before discussing the algorithm robustness, let us recall from chapter 1 the definition of Robust Schedule.

Robust schedule is the schedule which is invulnerable to little variation in activity durations caused by uncertainties (Chtourou & Haouari, 2008).

To be able to judge schedule robustness a measure of robustness is needed. In this discussion the deviation in the objective function value is the selected measure. Although more sophisticated robustness measures exist in the literature (The reader may refer to (Chtourou & Haouari, 2008) for more robustness measure), the selection of simple measure is preferred since the objective function is tangible measure and is easy to interpret.

Selecting the deviation in objective function value as a robustness measure falls under the category of quality robustness and not solution robustness as discussed earlier in chapter 2.

The methodology in increasing the solution robustness as explained chapter 3 was mainly based on selecting a schedule represented by completion time $\hat{\Theta}_{opt}$ (and it is associated resource assignment) that is expected to yield the least cost $\acute{Z}_{opt}$ considering uncertainties in the project activity durations.

Since the objective of RIP-TDRC is to find the optimal schedule leading to the least possible cost, it is clear that the optimal solution found by simulation ($\acute{Z}_{opt}$ associated with $\hat{\Theta}_{opt}$) is the most robust one, if we consider the deviation in objective function as robustness measure.

Now to see how the proposed algorithm has increased robustness let us see the results summarized in Table 6 below.

Table 6: Robustness comparision between the solution found by MIP model and the one found by simulation algorithm

	Optimal solution by MIP model Z_{opt}	**Average realized cost applying the schedule found by MIP Z_{act}**		**Average Realized cost applying the schedule found by simulation algorithm $\acute{Z}_{opt}$**		**Improvement**
		Z_{act}	**% deviation**	**$\acute{Z}_{opt}$**	**% deviation**	
A	1,698.15	2,463.96	45.10%	2,358.50	38.89%	4.28%
B	3,375.81	4,841.73	43.42%	4,679.50	38.62%	3.35%
C	5,433.15	7,768.85	42.99%	7,519.58	38.40%	3.21%
D	9,103.38	13,142.08	44.36%	12,853.69	41.20%	2.19%
E	1,679.04	2,767.88	64.85%	2,323.65	38.39%	16.05%
F	3,286.38	5,127.31	56.02%	4,556.73	38.65%	11.13%
G	5,381.77	8,349.88	55.15%	7,372.27	36.99%	11.71%
H	8,946.96	14,877.38	66.28%	12,477.85	39.46%	16.13%

Comparing the realized costs found after simulating the proposed schedules found by both MIP model (with a planned makespan of $\Theta_{\boldsymbol{opt}}$) and by our proposed algorithm (with a planned makespan of $\hat{\Theta}_{opt}$) , it has been found that the latter produces more robust schedules since its cost$\acute{Z}_{opt}$ has less deviation from the cost found by MIP model for the deterministic version $\boldsymbol{Z_{opt}}$ comparing to the first schedule.

The improvement on robustness varied between 2.19% and 16.13% depending on the project deadline for the reasons discussed in the coming subsections.

5.3 The effect of resource cost on the optimal project duration

In this section the effect on changing the resource cost assignment on the solution will be discussed. To study the effect of the resource cost assignment, each problem was solved with four different resource cost assignment sets from C1 to C4 discussed in the previous chapter.

It has been noticed that changing the resource cost does not have a major effect on shifting the optimal project makespan $\Theta_{\boldsymbol{opt}}$ and $\hat{\Theta}_{opt}$, in both problem variants (deterministic and stochastic).

The 26 instances were all solved with the same cost assignment C1 (represented by sub-instance suffix A) and the average of the makespans was calculated. Then the same was repeated for the other cost assignments (B to H). The results were grouped into two groups (A to D) and (E to H) to be able to find the effect of resource cost only independent from the project deadline.

Both Figure 7 and Figure 8 illustrate these findings for the deterministic version of the problem. Note that the variance for the first group is 0.0051 only, and 0.01368 for the second group.

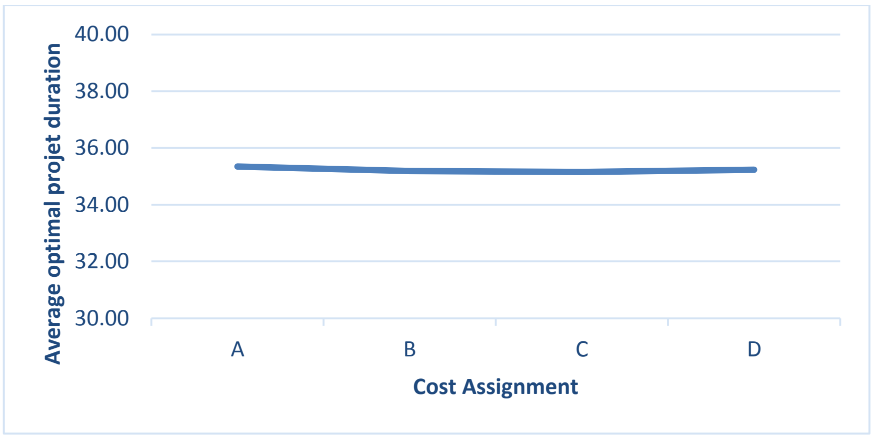

Figure 7: Effect of Resource Cost Assignment on Optimal Project Duration in the Deterministic Problem Variant- Chart 1

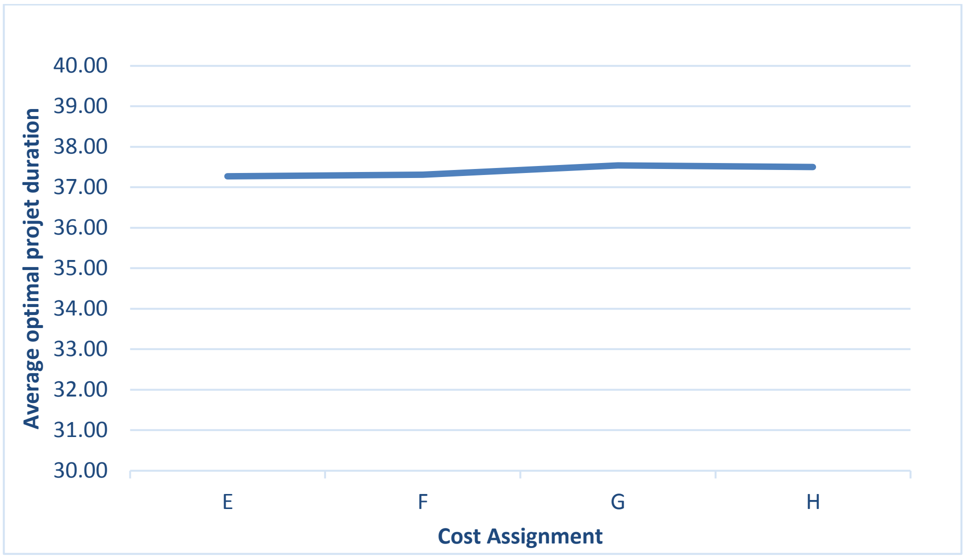

Figure 8: Effect of Resource Cost Assignment on Optimal Project Duration in the Deterministic Problem Variant- Chart 2

The same observation also applies on the stochastic version of the problem. The values of the selected (planned) project durations $\hat{\theta}_{opt}$ were affected much by the resource cost assignment. Refer to Figure 9 and Figure 10. The variance for the first group and second group was found to be 0.01368, and 0.01211 respectively.

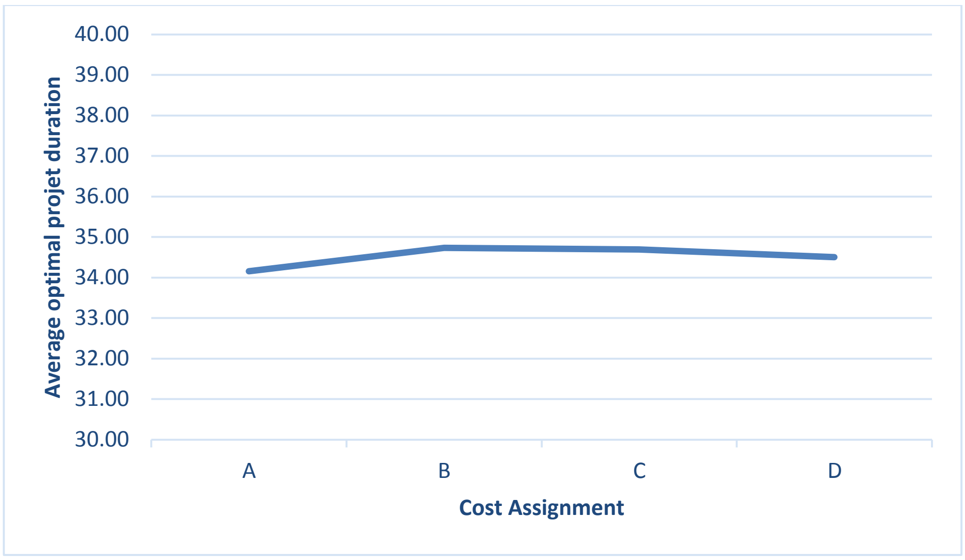

Figure 9: : Effect of Resource Cost Assignment on Optimal Project Duration in the Stochastic Problem Variant- Chart 1

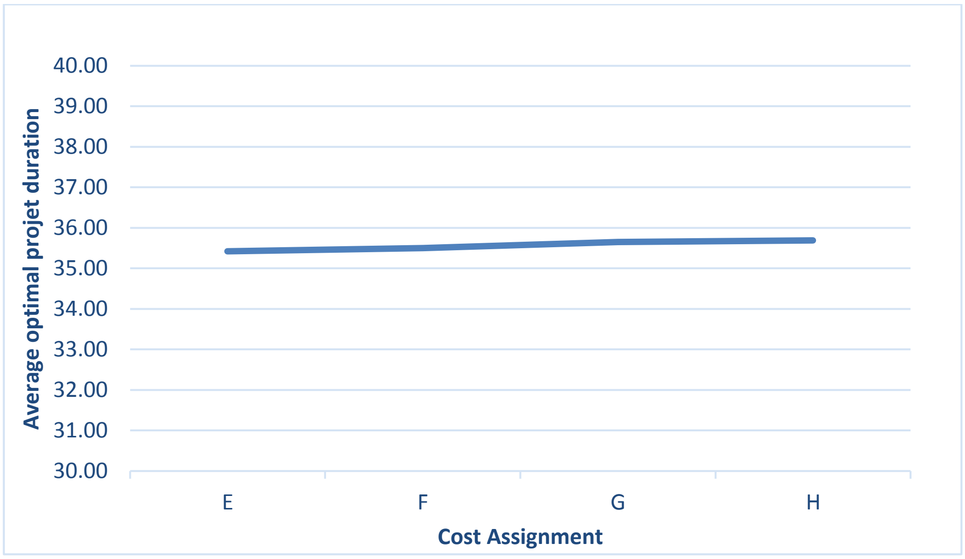

Figure 10: Effect of Resource Cost Assignment on Optimal Project Duration in the Stochastic Problem Variant- Chart 2

5.4 The effect of project deadline and penalty value on optimal project duration

To be able to analyze the effect of the project deadline and associated penalty (The penalty is a function of project deadline as explained in chapter 4) on the optimal project makespans $\Theta_{\boldsymbol{opt}}$ and $\hat{\Theta}_{opt}$, each problem instance was solved with two different project deadline for each cost assignment.

The results were grouped into four groups since we have solved for four different cost assignment. Group 1 will compare sub-instances A with E, while group 2 will compare sub-instances B with F and so on. The results on Tables 7 and 8can illustrate the effect of changing the project deadline on optimal solution.

Table 7: Effect of project deadline on the deterministic optimal solution

Sub-Instance	**Average Optimal Duration**		**Sub-Instance**	**Increase %**
Δ= min + (max-min) x 0.3		**Δ= min + (max-min) x 0.8**		
A	35.35	37.27	**E**	5.44%
B	35.19	37.31	**F**	6.01%
C	35.15	37.54	**G**	6.78%
D	35.23	37.50	**H**	6.44%

Table 8: Effect of project deadline on the non-deterministic optimal solution

Sub-Instance	**Average Optimal Duration**		**Sub-Instance**	**Increase %**
Δ= min + (max-min) x 0.3		**Δ= min + (max-min) x 0.8**		
A	34.15	35.42	**E**	3.72%
B	34.73	35.50	**F**	2.21%
C	34.69	35.65	**G**	2.77%
D	34.50	35.69	**H**	3.46%

It can be noticed that the increase in the project deadline resulted in increase on the optimal project makespan, for both of the problem variants. This result was expected, since increasing the project deadline will give the algorithm more freedom to find solutions with larger completion time, because the solution is now more bounded by the resource cost instead of being more bounded by tardiness cost.

It is also noted that for the stochastic version of the problem the percentage increase in the optimal project makespan is relatively smaller than the increase observed in the deterministic version. This result is expected as well and justified by what is discussed earlier in this chapter about the stochastic solution tendency to be shifted toward earlier completion time compared to the deterministic solution.

Also note that the increase in project deadline resulted in more gap between $\Theta_{\boldsymbol{opt}}$ and $\hat{\Theta}_{opt}$ (see table 5 in section 5.1) for the same reasons mentioned above. Since the increase in the exact optimal solution $\Theta_{\boldsymbol{opt}}$ due to the increase in project deadline was more than the increase noticed for $\hat{\Theta}_{opt}$, because of $\hat{\Theta}_{opt}$ tendency to be smaller than $\Theta_{\boldsymbol{opt}}$,. It is also worth mentioning that the gap between $\Theta_{\boldsymbol{opt}}$ and $\hat{\Theta}_{opt}$ in the sub-instances with larger project deadline made the algorithm more efficient in finding robust solutions. (Refer to Table 6 in section 5.2).

6 Conclusion and Future Work

In this thesis an extension of Resource Investment Problem (RIP) has been introduced to consider time-depended resource cost contrary to the classical RIP which deals only with time-independent resource cost. The problem was named Resource Investment Problem with Time-Dependent Resource Cost and Tardiness Penalty, abbreviated as (RIP-TDRC).

The thesis started with an introduction about project management problems, uncertainties and robustness. An extensive Literature Review about Resource Constrained Project Scheduling Problem (RCPSP) and other extensions of this problem was conducted. The Literature Review also discussed robust solutions techniques for RCPSP and RIP.

The new problem was also defined and a methodology to find exact optimal solution and robust simulation based solution was also introduced in this thesis.

Computational experiments were conducted, to validate the proposed algorithm, proof the solution robustness and to find the effects of some parameters on the problem solution such as, resource cost and project deadline.

The results of the experiment were discussed. The algorithm found valid and suitable for instances with small number of activities but showed noticeable deficiencies in larger instances, due to the use of exact MIP model in the algorithm.

The results also showed improvement in solution robustness if the algorithm is used compared to adapting the exact solution only.

The work done in this thesis can be extended in future by applying more robustness measures to assess the solution robustness and not only the quality robustness. The time performance of the algorithm can be improved by replacing the MIP model by any efficient heuristic. This will make the algorithm more capable to solve larger instances as well.

The same algorithm with some modification can be also used to solve other scheduling problems with different objective.

References

Al-Fawzan, M., & Haouari, M. (2005). A bi-objective model for robust resource-constrained project scheduling. *Int. J. Production Economics*, 175–187.

Alvarez-Valdés, R., & Tamarit, J. (1993). The Project Scheduling Polyhedron: Dimension, Facets and Lifting Theorems. *European Journal of Operational Research, 67*, 204-220.

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*, 297-316.

Artigues, C., Leus, R., & Nobibon, F. T. (2013). Robust optimization for resource-constrained project scheduling with uncertain activity durations. *Flex Serv Manuf J, 25*, 175–205.

Baar, T., Brucker, P., & Knust, S. (1998). Tabu-Search Algorithms for the Resource-Constrained Project Scheduling Problem. In S. Voss, I. O. S. Martello, & C.Roucairol, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization* (pp. 1-18). Dordrecht: Kluwer.

Cho, J.-H., & Kim, Y.-D. (1997). A simulated annealing algorithm for resource constrained project scheduling problems. *Journal of the Operational Research Society, 48*, 736-744.

Chtourou, H., & Haouari, M. (2008). A two-stage-priority-rule-based algorithm for robust resource constrained project scheduling. *Computers and Industrial Engineering*, 183-194.

Demeulemeester, & Herroelen. (2002). *Project Scheduling A Research Handbook.* Kluwer Academic Publishers.

Demeulemeester, E., & Herroelen, W. (n.d.). *The Library PSBLIB.* Retrieved from http://www.om-db.wi.tum.de/psplib/library.html

Drexl, A., & Kimms, A. (2001). Optimization Guided Lower and Upper Bounds for the Resource Investment Problem. *The Journal of the Operational Research Society, 52*(3), 340-351.

Glover, F. (1989). Tabu search Part I. *ORSA Journal on Computing*, 190{206.

Hartmann, S. (1998). A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling. *Naval Research Logistics, 45*, 733-750.

Hartmann, S., & Briskorn, D. (2009). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*.

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

Herroelen, W., & Leus, R. (2004). The construction of stable project baseline schedules. *European Journal of Operational Research, 156*, 550–565.

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

Hillier, S. F., & Lieberman, G. J. (2008). *Introduction to Operations Research* (Ninth ed.). Mc Graw. Hill.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, J. M. (1983). Optimization by Simulated Annealing. *Science, 220*, 671-680.

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

Kolisch, R., & Hartmann, S. (1999). Heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis. Christian Albrechts Universityl.

Kolisch, R., & Hartmann, S. (1999). Heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis. Christian Albrechts University.

Kone´, O., Artigues, C., Lopez, P., & Mongeau, M. (2011). Event-based MILP models for resource-constrained project scheduling Problem. *Computers & Operations Research, 38*, 3-13.

Lee, J., & Kim, Y. (1996). Search Heuristics for Resource Constrained Project Scheduling. *Journal of The Operations Research Society, 47*, 678-689.

Leon, V. J., & Balakrishnan, R. (1995). Strength and adaptability of problem-space based neighborhoods for resource-constrained scheduling. *ORSpektrum, 17*, 173-182.

Leus, R. (2003). *The generation of stable project plans Complexity and exact algorithms.* LEUVEN: KATHOLIEKE UNIVERSITEIT LEUVEN.

Leus, R., & Herroelen, W. (2004). Stability and resource allocation in project planning. *IIE Transactions, 36*, 667–682.

Magalhães-Mendes, J. (2011). Active, Parameterized Active, and Non-Delay Schedules. *Recent Researches in Applied Mathematics and Informatics*.

Mingozzi, A., Maniezzo, V., Ricciardelli, S., & Bianco, L. (1988). An Exact Algorithm for the Resource-Constrained Project Scheduling Problem Based on a New Mathematical Formulation. *Management Science, 44*, 715-729.

Najafi, A. A., & Azimi, F. (2009). A Priority Rule-Based Heuristic for Resource Investment Project Scheduling Problem with Discounted Cash Flows and Tardiness Penalties. *Mathematical Problems in Engineering*.

Nubel, H. (2001). The resource renting problem subject to temporal constraints. *OR Spektrum , 23,* 359-381.

PMI. (2013). *A Guide to the Project Management Body of Knowledge (PMBOK Guide)- 5th Edition.* Newtown Square, pennsylvania, USA: Project Management Institute.

Pritsker, A., Watters, L., & P.Wolfe. (1969). MULTIPROJECT SCHEDULING WITH LIMITED RESOURCES: A ZERO-ONE PROGRAMMING APPROACH. *MANAGEMENT SCIENCE, 16*.

Ranjbar, M., Kianfar, F., & Shadrokh, S. (2008). Solving the resource availability cost problem in project scheduling by path relinking and genetic algorithm. *Applied Mathematics and Computation, 196*, 879-888.

Schwindt, C., & Zimmermann, J. (2015). *Handbook on project managment and scheduling*. Springer.

Shadrokh, S., & Kianfar, F. (2007). A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty. *European Journal of Operational Research, 181*, 86-101.

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*, 723–733.

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. *Int. J. Production Economics, 97*, 227–240.

Van De Vonder, S., Demeulemeester, E., Herroelen, W., & Leus, R. (2006). The trade-off between stability and makespan in resource-constrained project scheduling. *International Journal of Production Research, 44*, 215–236.

Xiong, J., Chen, Y.-w., Yang, K.-w., Zhao, Q.-s., & Xing, L.-n. (2012). A Hybrid Multi-objective Genetic Algorithm for Robust Resource-Constrained Project Scheduling with Stochastic Durations. *Mathematical Problems in Engineering*.

Yamashita, D. S., Armentano, V. A., & Laguna, M. (2007). Robust optimization models for project scheduling with resource availability cost. *Journal of Scheduling, 10*, 67-76.

Appendices

Appendix A: 20 non-dummy activity instances

Instance Number:		202-2			Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	8	4	0	0	0	5	10	14
2	4	10	0	0	0	6	7	12
3	6	0	0	0	3	4	8	9
4	3	3	0	0	0	19		
5	8	0	0	0	8	21		
6	5	4	0	0	0	21		
7	9	0	1	0	0	11	18	
8	2	6	0	0	0	13		
9	7	0	0	0	1	15		
10	9	0	5	0	0	19		
11	2	0	7	0	0	13		
12	6	4	0	0	0	16	17	
13	3	0	8	0	0	16		
14	9	3	0	0	0	21		
15	10	0	0	0	5	20	21	
16	6	0	0	0	8	21		
17	5	0	0	0	7	19		
18	3	0	1	0	0	21		
19	7	0	10	0	0	21		
20	2	0	0	0	6	21		
21	0	0	0	0	0			

Instance Number:		203-2			Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	6	0	0	2	0	5	11	17
2	9	9	0	0	0	21		
3	5	0	0	2	0	4	9	
4	7	0	0	0	6	6	18	20
5	5	0	3	0	0	7	8	13
6	9	0	6	0	0	14	17	
7	9	0	0	0	9	10		
8	9	0	0	0	6	12		
9	1	4	0	0	0	21		
10	1	0	0	0	6	15		
11	6	0	3	0	0	16		
12	5	0	9	0	0	15		
13	3	0	8	0	0	16		
14	3	0	0	0	2	21		
15	2	0	7	0	0	21		
16	5	0	0	5	0	21		
17	4	0	0	0	8	21		
18	8	0	0	0	1	19		
19	8	0	0	10	0	21		
20	2	0	3	0	0	21		
21	0	0	0	0	0			

Instance Number:		205-1			Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	10	5	0	0	8	11		
2	4	10	0	5	0	4	7	16
3	1	5	0	1	0	5	18	
4	3	0	0	10	5	6	9	
5	5	3	0	0	0	15		
6	10	0	0	5	0	8		
7	1	0	4	0	0	10		
8	4	5	0	0	7	21		
9	6	1	4	4	8	12		
10	8	2	0	0	1	14		
11	7	0	10	7	6	21		
12	7	10	0	0	5	13		
13	4	0	6	0	0	19		
14	3	9	0	3	10	21		
15	10	3	5	0	0	20		
16	3	10	4	3	10	17		
17	4	0	1	0	1	21		
18	3	6	6	0	1	19	20	
19	7	0	6	0	0	21		
20	5	0	0	5	0	21		
21	0	0	0	0	0			

Instance Number:		205-2			Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	9	0	4	0	10	4	6	10
2	5	0	2	0	9	7		
3	9	1	10	0	0	8	14	20
4	8	10	1	9	7	5	15	19
5	6	0	0	0	5	21		
6	3	9	0	0	10	12	13	
7	10	0	0	0	3	9	14	
8	10	0	5	2	0	11		
9	8	0	0	9	0	16	18	
10	3	1	0	2	0	13		
11	9	4	0	0	0	21		
12	1	0	0	0	4	15	21	
13	7	0	2	7	0	21		
14	1	5	0	3	9	17		
15	4	5	7	8	8	16		
16	7	0	5	8	0	21		
17	1	1	5	0	0	21		
18	2	1	0	7	0	21		
19	1	2	0	1	0	21		
20	8	4	0	9	0	21		
21	0	0	0	0	0			

Instance Number:	206-1				Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	10	4	0	0	7	5		
2	10	10	0	0	0	4		
3	4	5	0	0	0	6	19	
4	9	9	5	0	0	7	10	19
5	5	4	4	7	9	13	14	16
6	7	9	3	0	5	9	14	
7	2	0	0	8	0	8		
8	7	1	2	0	0	17		
9	2	9	10	0	7	10	12	
10	3	0	0	0	10	11		
11	7	7	9	10	3	21		
12	5	0	0	0	8	21		
13	2	0	0	4	0	15	17	
14	7	0	0	10	7	21		
15	9	0	10	0	0	20		
16	3	0	0	4	1	18		
17	7	4	4	0	7	18		
18	5	9	5	0	0	21		
19	3	0	7	10	0	21		
20	6	4	1	0	0	21		
21	0	0	0	0	0			

Instance Number:	206-2				Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	4	4	10	10	0	5	7	11
2	6	0	3	8	2	6		
3	1	1	0	0	5	4		
4	6	0	6	8	0	13	15	18
5	9	0	10	9	9	8	9	
6	3	1	5	0	3	21		
7	7	0	0	0	1	20		
8	2	4	0	0	5	10		
9	8	8	0	0	0	12		
10	7	0	0	7	1	21		
11	7	10	0	0	7	21		
12	2	4	10	0	2	14		
13	8	8	0	3	0	19		
14	5	0	5	0	0	16	17	
15	4	0	8	0	0	21		
16	1	6	4	0	0	20		
17	10	9	0	5	4	21		
18	9	6	0	9	0	20		
19	8	0	0	0	9	21		
20	6	1	8	9	0	21		
21	0	0	0	0	0			

Instance Number:	207-1				Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	6	2	0	10	0	5		
2	2	9	0	3	2	16		
3	8	2	3	0	0	4	7	8
4	2	0	8	0	0	10	20	
5	6	4	5	0	1	6	14	
6	4	1	0	7	0	9	19	
7	5	0	0	0	4	11	18	
8	3	9	10	5	2	21		
9	10	0	0	2	0	15	17	
10	2	1	8	1	0	13		
11	7	1	0	1	0	12		
12	4	0	0	3	3	17		
13	3	6	0	8	0	17		
14	9	0	10	2	7	18		
15	3	0	8	4	0	21		
16	6	0	7	0	0	21		
17	1	8	0	0	0	21		
18	3	2	0	0	6	21		
19	6	0	1	0	0	21		
20	1	0	8	0	2	21		
21	0	0	0	0	0			

Instance Number:	207-2			Number of Resource types: 4				
Number nun-dummy of activities:20				Maximum # of Successors: 3				
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	8	6	0	0	0	6	18	
2	7	10	0	4	6	4	8	
3	3	8	0	0	8	9	16	
4	7	0	10	5	3	5	7	
5	3	0	9	0	10	17	19	
6	9	0	0	0	4	20		
7	3	0	0	0	3	11		
8	9	0	6	0	0	12		
9	4	0	10	8	10	10	13	15
10	3	7	0	0	8	21		
11	9	0	10	9	0	21		
12	4	6	0	0	0	14		
13	10	5	9	0	5	20		
14	7	1	2	0	7	21		
15	2	0	5	0	7	21		
16	8	10	0	1	7	21		
17	6	7	0	0	0	21		
18	1	2	5	10	0	20		
19	9	0	0	0	3	21		
20	2	0	8	4	8	21		
21	0	0	0	0	0			

Instance Number:	208-1	Number of Resource types: 4
Number nun-dummy of activities:20		Maximum # of Successors: 3

Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	6	1	0	0	2	6		
2	9	7	7	3	0	7	12	
3	7	10	0	10	3	4	5	17
4	1	1	0	8	9	10	14	
5	2	0	0	5	9	8	11	16
6	5	5	0	10	0	9	19	
7	7	0	0	3	9	21		
8	7	0	5	1	0	15		
9	7	7	2	0	0	21		
10	1	8	0	3	10	12		
11	3	5	6	0	0	21		
12	1	0	7	0	8	13	18	
13	8	8	0	10	4	21		
14	1	9	0	0	0	21		
15	7	0	0	0	9	21		
16	1	0	0	10	0	18		
17	2	3	7	0	0	21		
18	6	0	0	0	8	20		
19	4	0	9	5	0	21		
20	9	0	6	0	0	21		
21	0	0	0	0	0			

Instance Number:	208-2				Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	6	5	0	0	0	4	7	
2	5	5	2	8	10	8	10	19
3	3	8	0	0	0	8	11	
4	3	3	0	5	0	5		
5	8	7	6	0	0	6		
6	6	1	0	1	0	21		
7	8	0	7	0	9	13	14	
8	8	0	0	9	0	9	12	
9	6	0	8	2	0	18		
10	10	9	0	0	0	11	20	
11	8	0	0	2	7	21		
12	8	5	2	2	0	15	16	
13	6	0	0	0	5	17		
14	5	10	7	2	4	17	21	
15	2	8	3	0	1	17	20	
16	9	0	0	0	9	21		
17	7	5	5	0	6	21		
18	5	0	4	0	1	21		
19	4	2	0	8	0	21		
20	3	0	6	0	5	21		
21	0	0	0	0	0			

Instance Number:		2017-1			Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	5	0	0	0	9	4	11	14
2	2	9	0	0	0	12	13	
3	5	0	0	0	3	5	7	19
4	8	0	0	0	10	9		
5	8	0	0	7	0	6	8	15
6	9	0	0	0	10	9	17	
7	6	0	0	8	0	17		
8	1	10	0	0	0	10	16	18
9	10	0	0	10	0	21		
10	2	0	0	5	0	17		
11	2	0	9	0	0	13	18	
12	7	0	5	0	0	20		
13	5	0	2	0	0	21		
14	10	0	0	0	9	15		
15	9	0	8	0	0	18		
16	8	0	0	6	0	21		
17	9	0	0	0	6	21		
18	10	0	0	0	8	21		
19	8	0	0	10	0	21		
20	6	0	10	0	0	21		
21	0	0	0	0	0			

Instance Number:	2018-1				Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	8	0	0	0	3	4	9	10
2	4	0	1	0	0	4	7	14
3	9	0	0	0	4	6		
4	9	8	0	0	0	5	15	17
5	2	0	0	0	2	21		
6	5	0	0	4	0	11	14	19
7	1	0	3	0	0	8	16	
8	2	0	0	0	5	13		
9	6	0	10	0	0	12		
10	9	4	0	0	0	21		
11	1	0	0	9	0	13	15	
12	10	7	0	0	0	18		
13	10	0	9	0	0	21		
14	5	2	0	0	0	16	20	
15	7	0	10	0	0	21		
16	9	7	0	0	0	21		
17	4	0	0	8	0	19	21	
18	1	0	0	0	1	19		
19	4	0	0	2	0	21		
20	2	0	0	0	9	21		
21	0	0	0	0	0			

Instance Number:		2019-1			Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	5	0	3	0	0	4	6	12
2	1	7	0	0	0	10		
3	6	9	0	0	0	5	7	13
4	8	0	0	1	0	10	15	
5	5	0	0	0	3	10	17	
6	1	8	0	0	0	8	15	18
7	4	0	0	1	0	9	11	
8	10	0	0	9	0	19		
9	7	0	9	0	0	17	18	
10	3	6	0	0	0	21		
11	1	0	0	5	0	14		
12	10	0	0	8	0	20		
13	4	0	0	2	0	19	20	
14	4	0	7	0	0	19		
15	4	0	5	0	0	16		
16	5	0	10	0	0	21		
17	7	4	0	0	0	21		
18	3	7	0	0	0	21		
19	7	0	10	0	0	21		
20	3	1	0	0	0	21		
21	0	0	0	0	0			

Instance Number:	2019-2	Number of Resource types: 4
Number nun-dummy of activities:20		Maximum # of Successors: 3

Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	5	0	0	8	0	4	5	6
2	6	0	0	0	5	7	16	19
3	8	5	0	0	0	10	15	
4	6	0	0	0	4	8	18	
5	5	0	0	4	0	17		
6	4	0	0	1	0	17		
7	6	0	0	6	0	9	14	
8	5	0	3	0	0	11	12	13
9	8	0	0	0	3	10		
10	10	0	0	0	1	21		
11	8	0	0	0	8	21		
12	9	0	0	0	4	15		
13	7	0	0	0	1	19		
14	3	0	0	2	0	15	18	
15	9	0	0	1	0	17	20	
16	1	0	10	0	0	21		
17	1	0	7	0	0	21		
18	6	0	0	0	1	20		
19	6	0	0	0	8	21		
20	6	5	0	0	0	21		
21	0	0	0	0	0			

Instance Number:		2020-1			Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	7	0	0	0	10	7		
2	8	9	0	0	0	9	11	
3	8	0	7	0	0	4	5	14
4	7	0	0	9	0	6	10	
5	6	0	0	0	1	8	17	
6	7	0	0	0	2	16		
7	8	0	0	7	0	8	20	
8	7	0	2	0	0	19		
9	10	0	0	2	0	13	15	
10	5	0	0	0	1	12	13	
11	3	0	0	6	0	12		
12	3	0	6	0	0	21		
13	1	4	0	0	0	21		
14	1	0	0	7	0	17	18	
15	5	0	0	0	9	16	19	
16	9	0	1	0	0	17	18	
17	5	0	0	0	7	20		
18	1	7	0	0	0	21		
19	3	0	0	0	5	21		
20	9	0	0	3	0	21		
21	0	0	0	0	0			

Instance Number:		2022-1			Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	2	9	0	8	0	6	8	
2	2	3	0	3	8	8		
3	10	0	7	10	2	4	16	21
4	6	7	0	0	3	5		
5	2	0	8	0	2	14		
6	5	0	0	8	7	7	9	10
7	2	5	3	7	0	12		
8	8	7	0	0	7	11	13	
9	9	2	0	7	2	19		
10	6	4	0	3	0	15	19	
11	4	0	3	0	0	15	17	
12	1	10	0	1	0	13	16	18
13	2	0	6	3	1	14	17	
14	7	0	0	9	1	21		
15	4	0	2	2	0	21		
16	7	0	0	5	4	21		
17	1	0	9	0	10	21		
18	7	4	0	0	0	20		
19	4	0	6	0	0	21		
20	3	0	0	10	8	21		
21	0	0	0	0	0			

Instance Number:	2023-1				Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	10	1	0	1	0	5	13	
2	4	4	0	0	3	4	12	
3	5	0	2	6	4	6	8	9
4	1	0	0	7	5	11		
5	5	8	4	7	6	7	9	16
6	9	8	6	0	0	15	18	
7	3	0	0	7	7	18		
8	9	0	0	10	8	10		
9	7	0	2	0	0	10		
10	9	1	10	0	0	14	17	
11	8	4	0	0	6	17		
12	1	0	6	0	1	21		
13	7	2	0	0	0	15		
14	9	4	0	0	0	20		
15	5	0	0	0	10	19		
16	4	2	3	0	9	21		
17	2	0	0	9	7	21		
18	8	8	0	0	0	20		
19	6	8	9	1	10	21		
20	2	8	7	0	1	21		
21	0	0	0	0	0			

Instance Number:		2023-2			Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	3	0	0	0	9	9	14	
2	1	0	2	0	0	10	12	16
3	3	10	5	0	9	4	5	6
4	1	2	0	2	3	15	17	19
5	6	5	3	1	0	7	8	11
6	2	1	9	3	7	9		
7	9	0	6	10	0	15		
8	5	0	0	6	0	13		
9	8	7	4	0	1	10		
10	10	0	3	0	7	20		
11	2	0	6	0	7	15	18	
12	3	8	0	9	0	21		
13	5	0	6	5	0	21		
14	1	0	4	0	0	20		
15	7	0	0	2	7	21		
16	7	0	0	0	2	19		
17	7	2	0	0	0	21		
18	4	0	7	2	0	19	20	
19	1	5	9	6	0	21		
20	7	10	0	0	10	21		
21	0	0	0	0	0			

Instance Number:		2024-1			Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	9	1	0	2	0	7		
2	7	4	0	0	5	4	5	6
3	1	1	3	0	0	8	11	18
4	8	5	2	0	8	7	8	
5	10	3	0	0	0	10	19	
6	8	4	0	3	7	19		
7	3	0	3	8	0	9	13	20
8	7	0	1	0	10	9	20	
9	4	8	0	5	0	12	14	
10	1	0	7	0	6	11	14	16
11	6	0	0	2	4	21		
12	7	1	3	8	0	15	17	
13	3	8	3	0	0	15		
14	1	0	0	2	2	15		
15	2	0	7	4	6	21		
16	1	0	0	1	9	21		
17	6	0	7	4	6	21		
18	6	0	0	0	2	19		
19	4	0	10	0	0	21		
20	10	7	1	9	0	21		
21	0	0	0	0	0			

Instance Number:	2024-2	Number of Resource types: 4
Number nun-dummy of activities:20		Maximum # of Successors: 3

Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	4	5	0	0	0	8	20	
2	3	0	0	0	3	6	10	12
3	10	7	0	0	1	4	5	16
4	4	10	0	0	10	7	9	11
5	9	3	2	7	4	15	17	20
6	2	0	0	3	0	21		
7	8	0	8	1	0	15	17	
8	6	3	0	0	0	21		
9	8	10	0	9	7	13		
10	5	9	3	5	4	14	19	
11	1	0	2	9	0	14	18	
12	3	3	0	0	10	18		
13	3	0	1	0	2	21		
14	1	8	0	0	0	21		
15	5	10	0	7	0	18		
16	6	0	0	2	6	17		
17	8	6	10	0	9	19		
18	5	0	4	1	9	21		
19	7	4	0	0	1	21		
20	1	9	0	9	0	21		
21	0	0	0	0	0			

Instance Number:	2033-1				Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	3	0	0	0	9	11	15	
2	8	0	0	0	7	7	12	
3	4	0	7	0	0	4	5	11
4	4	0	3	0	0	13	17	
5	6	0	0	9	0	6	7	
6	5	0	0	3	0	12	17	
7	7	10	0	0	0	8	9	18
8	2	0	0	7	0	10	13	14
9	5	0	0	7	0	14	19	
10	1	1	0	0	0	16		
11	9	0	0	5	0	12	16	18
12	9	0	0	0	5	13	19	
13	5	9	0	0	0	21		
14	10	0	0	0	3	20		
15	8	0	0	8	0	18	20	
16	10	0	0	0	1	17		
17	2	0	0	0	7	20		
18	8	0	0	0	4	19		
19	8	0	5	0	0	21		
20	9	0	4	0	0	21		
21	0	0	0	0	0			

Instance Number:	2033-2	Number of Resource types: 4
Number nun-dummy of activities:20		Maximum # of Successors: 3

Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	6	0	0	0	2	10	19	
2	6	0	0	0	8	5	9	
3	7	0	0	0	2	4	6	9
4	6	4	0	0	0	7	8	20
5	10	0	0	8	0	10	20	
6	7	0	6	0	0	11	12	20
7	10	0	0	0	9	14	15	18
8	4	0	7	0	0	10	12	
9	6	2	0	0	0	13	19	
10	8	1	0	0	0	17		
11	8	0	4	0	0	13	16	
12	8	1	0	0	0	18		
13	4	0	0	0	7	14	15	
14	2	10	0	0	0	17		
15	5	2	0	0	0	21		
16	2	0	1	0	0	17	18	
17	8	0	0	0	4	21		
18	1	0	0	6	0	19		
19	2	0	0	1	0	21		
20	7	0	8	0	0	21		
21	0	0	0	0	0			

Instance Number:		2035-2			Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	7	0	2	0	0	6	11	
2	4	0	0	4	0	4	10	15
3	1	0	0	7	0	5	7	12
4	5	0	0	0	4	12	14	
5	9	0	0	6	0	9	14	
6	7	0	0	5	0	7	15	16
7	5	0	0	0	7	8	13	17
8	7	0	0	2	0	19	20	
9	10	0	0	0	2	16	18	
10	8	0	0	0	9	13		
11	2	0	10	0	0	12	17	
12	6	0	8	0	0	16		
13	2	0	0	5	0	14		
14	6	0	4	0	0	21		
15	5	0	0	0	8	17		
16	7	5	0	0	0	20		
17	4	10	0	0	0	19		
18	9	0	0	0	5	19	20	
19	3	6	0	0	0	21		
20	7	0	0	2	0	21		
21	0	0	0	0	0			

Instance Number:	2036-1	Number of Resource types: 4
Number nun-dummy of activities:20		Maximum # of Successors: 3

Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	6	8	0	0	0	8	20	
2	3	0	0	0	2	4	5	7
3	5	0	0	0	1	6	9	11
4	6	7	0	0	0	10	16	18
5	10	6	0	0	0	6	9	14
6	5	6	0	0	0	18	20	
7	2	0	0	7	0	8	11	17
8	10	0	0	0	4	16		
9	10	0	0	0	10	13	19	
10	7	0	0	10	0	12	13	19
11	10	0	0	0	8	15	16	
12	3	4	0	0	0	15		
13	6	0	0	0	9	17	20	
14	8	5	0	0	0	15		
15	10	0	0	2	0	21		
16	1	1	0	0	0	21		
17	1	8	0	0	0	21		
18	1	0	0	8	0	19		
19	8	0	8	0	0	21		
20	3	0	0	0	10	21		
21	0	0	0	0	0			

Instance Number:		2036-2			Number of Resource types: 4			
Number nun-dummy of activities:20					Maximum # of Successors: 3			
Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	3	8	0	0	0	4	5	
2	10	5	0	0	0	21		
3	2	0	0	5	0	8	12	15
4	5	9	0	0	0	7	10	12
5	10	2	0	0	0	6	13	
6	1	6	0	0	0	11	18	
7	1	0	0	4	0	14	16	17
8	10	7	0	0	0	9	11	
9	3	0	0	0	3	13	16	20
10	9	1	0	0	0	11	13	14
11	5	0	0	0	1	16	17	
12	10	0	0	0	8	14	20	
13	5	0	1	0	0	17	18	
14	4	0	1	0	0	18		
15	4	0	0	0	5	19	20	
16	6	0	0	9	0	21		
17	6	8	0	0	0	21		
18	6	0	1	0	0	21		
19	8	0	0	0	4	21		
20	1	0	0	8	0	21		
21	0	0	0	0	0			

Instance Number:	2036-2	Number of Resource types: 4
Number nun-dummy of activities:20		Maximum # of Successors: 3

Activity ID	Duration	R1	R2	R3	R4	Suc 1	Suc 2	Suc 3
0	0	0	0	0	0	1	2	3
1	2	0	9	0	0	4	5	6
2	7	0	0	2	0	9	13	14
3	9	8	6	0	3	4	6	7
4	7	2	9	2	0	11	12	16
5	6	8	0	2	0	10	12	17
6	2	0	8	0	0	8	13	
7	2	0	10	7	0	13	21	
8	8	0	3	0	0	11	20	
9	6	1	0	0	10	10		
10	8	10	6	0	6	15	16	
11	2	7	3	0	0	15		
12	1	9	0	0	3	15	18	
13	9	9	0	3	9	20		
14	3	0	4	0	2	19	21	
15	7	0	9	5	0	21		
16	2	0	0	2	0	18	19	
17	3	0	0	5	9	18	20	
18	7	3	3	0	7	21		
19	8	0	2	0	9	21		
20	6	4	0	0	0	21		
21	0	0	0	0	0			

Appendix B: Summary of Experiments Results

Instance Number	Sub Instance	Θ_{opt}	$\hat{\Theta}_{opt}$	Θ_{act}	$\boldsymbol{Z}_{opt}$	$\acute{Z}_{opt}$	$\boldsymbol{Z}_{act}$
J202-2	**A**	31	29	34	1116	1592	1631
	B	29	29	34	2146	2957	2957
	C	29	29	34	3393	4700	4700
	D	29	29	34	5684	7888	7888
	E	31	29	34	1116	1389	1431
	F	29	29	34	2146	2572	2572
	G	29	29	34	3393	4062	4062
	H	29	29	34	5684	6804	6804
J203-2	**A**	31	28	33	1364	1835	2039
	B	28	29	33	2940	3694	3695
	C	28	28	33	4788	6330	6330
	D	28	29	33	8260	10973	10988
	E	35	29	33	1330	1629	3177
	F	35	29	33	2695	3492	6401
	G	35	29	33	4410	5686	10565
	H	35	29	33	7525	9805	18018
J205-1	**A**	31	31	38	1767	2681	2681
	B	31	32	38	3565	5237	5401
	C	34	32	38	5644	8386	9256
	D	34	32	38	9452	14260	15086
	E	38	32	39	1748	2469	6036
	F	38	32	38	3564	4437	5858
	G	37	32	38	5587	7073	11848
	H	35	32	38	9170	12022	16109
J205-2	**A**	34	32	33	2124	2700	2798
	B	36	38	43	4212	5644	6042
	C	36	37	43	6498	9145	10010
	D	36	36	44	11736	15991	15991
	E	42	37	44	2100	2607	3107
	F	42	37	43	4200	5091	5509
	G	36	37	43	6948	8311	8795
	H	36	36	44	11736	14397	14397

Instance Number	Sub Instance	Θ_{opt}	$\hat{\Theta}_{opt}$	Θ_{act}	$\boldsymbol{Z}_{opt}$	$\acute{Z}_{opt}$	$\boldsymbol{Z}_{act}$
J206-1	**A**	46	40	46	2418	2975	3613
	B	46	40	46	4457	5599	6633
	C	41	40	46	6888	8679	9216
	D	41	42	48	11193	14562	15040
	E	46	40	46	2346	2853	3262
	F	46	40	46	4324	5347	5986
	G	46	40	46	6762	8298	9461
	H	46	42	48	11040	13292	15560
J206-2	**A**	45	40	46	2250	2847	3034
	B	45	40	46	4230	5573	5876
	C	45	41	49	6930	9371	9632
	D	45	41	49	11565	15871	16124
	E	48	46	53	2160	2590	2784
	F	46	46	53	4186	4912	4912
	G	46	47	54	6854	8048	8060
	H	46	46	54	11454	13467	13467
J207-1	**A**	31	29	35	1333	2220	2278
	B	31	29	34	2542	3860	4652
	C	31	29	34	4123	6195	7423
	D	31	29	34	6727	10119	11552
	E	33	29	35	1287	2327	3991
	F	32	29	34	2496	3217	4886
	G	32	29	34	4000	5148	9075
	H	32	29	34	6496	8445	18182
J207-2	**A**	28	28	34	1988	2440	2440
	B	28	28	34	4004	4867	4867
	C	28	29	34	6384	7723	7758
	D	30	30	35	10680	12938	12938
	E	28	28	34	1988	2436	2436
	F	28	28	34	4004	4871	4871
	G	28	29	34	6384	7721	7764
	H	30	30	35	10680	12763	12763
J208-1	**A**	26	25	30	1742	2195	2426
	B	26	25	30	3483	4366	4368
	C	27	26	30	5616	6807	7091
	D	27	27	31	9531	11879	11879
	E	32	29	34	1728	2092	2336
	F	32	27	31	3328	4199	4277
	G	32	29	33	5344	6586	6858
	H	32	27	31	9024	11291	11338

Instance Number	Sub Instance	Θ_{opt}	$\hat{\Theta}_{opt}$	Θ_{act}	Z_{opt}	$\acute{Z}_{opt}$	Z_{act}
J208-2	A	35	34	40	1995	2615	2898
	B	33	34	41	3894	5258	5301
	C	33	34	41	6039	8135	8298
	D	33	34	41	10065	13578	13741
	E	42	37	43	1974	2335	3072
	F	39	40	47	3783	4606	5940
	G	39	40	47	5928	7235	9397
	H	39	42	48	9945	12135	17017
J2017-1	A	34	34	39	1870	2250	2251
	B	34	34	39	4046	4871	4871
	C	34	35	39	6766	8142	8145
	D	34	34	39	11968	14343	14343
	E	34	34	39	1925	2193	2193
	F	34	35	39	4046	4745	4749
	G	34	34	39	6766	7925	7925
	H	34	35	39	11968	14024	14039
j2018-1	A	32	32	37	1344	1777	1777
	B	32	32	37	2528	3348	3348
	C	32	33	37	4000	5298	5302
	D	32	29	33	6624	8435	8758
	E	32	32	37	1344	1590	1590
	F	32	33	38	2528	2991	2993
	G	32	33	38	4000	4730	4740
	H	32	33	37	6624	7835	7837
J2019-1	A	24	25	28	1248	1537	1568
	B	24	25	28	2304	2841	2901
	C	24	25	28	3792	4659	4754
	D	24	25	28	5928	7311	7448
	E	33	25	28	1155	1497	2182
	F	34	24	29	2108	2764	4097
	G	34	27	31	3400	4548	6750
	H	34	25	28	5338	7190	10586
J2019-2	A	40	41	47	1280	2538	2544
	B	40	43	47	2720	5384	5403
	C	40	40	47	4480	8889	8889
	D	40	41	47	7640	15165	15214
	E	40	40	47	1280	3587	3587
	F	40	42	47	2720	7603	7663
	G	40	41	47	4480	12462	12554
	H	40	42	47	7640	21284	21441

Instance Number	**Sub Instance**	θ_{opt}	$\hat{\theta}_{opt}$	θ_{act}	$\boldsymbol{Z}_{opt}$	$\acute{Z}_{opt}$	$\boldsymbol{Z}_{act}$
J2020-1	**A**	46	46	54	1748	2953	2953
	B	46	47	54	3588	6060	6080
	C	46	49	54	5934	10073	10077
	D	46	46	54	10304	17362	17362
	E	46	48	54	1748	3716	3750
	F	46	47	54	3588	7590	7686
	G	46	47	54	5934	12704	12710
	H	46	48	54	10304	21884	22237
J2022-1	**A**	26	26	30	1566	1906	2016
	B	26	26	30	3042	3665	3665
	C	26	26	30	5044	6064	6064
	D	26	26	30	5828	10249	10249
	E	27	27	32	1566	1864	1864
	F	26	26	30	3042	3583	3583
	G	26	26	30	5044	5925	5925
	H	26	27	31	5828	10036	10050
J2023-1	**A**	45	47	53	1980	2896	3038
	B	45	47	53	3960	5823	6091
	C	45	42	48	6390	9655	9858
	D	45	42	48	10890	16518	17021
	E	45	47	53	1980	2552	2912
	F	48	47	43	3936	5089	7481
	G	48	44	53	6336	8778	11974
	H	48	44	53	10656	14814	20300
J2023-2	**A**	34	30	35	1589	2171	2323
	B	34	30	34	3332	4360	4875
	C	34	30	35	5406	6927	7765
	D	34	30	35	9214	11753	12437
	E	34	36	42	1598	1897	1933
	F	40	35	41	3280	3960	4476
	G	40	35	41	5280	6444	6711
	H	40	34	39	8880	10728	19970
J2024-1	**A**	46	44	50	1987	2785	2961
	B	45	45	52	4050	5611	5611
	C	45	43	50	6615	9051	9190
	D	45	44	50	11115	15200	15456
	E	46	45	53	1932	2400	2624
	F	46	45	52	3956	4744	5402
	G	46	45	52	6578	7762	7867
	H	45	45	52	11115	13040	13040

Instance Number	Sub Instance	Θ_{opt}	$\hat{\Theta}_{opt}$	Θ_{act}	Z_{opt}	$\acute{Z}_{opt}$	Z_{act}
J2024-2	**A**	38	38	44	2014	2573	2573
	B	38	38	44	4066	5121	5121
	C	38	38	44	6498	8224	8224
	D	39	38	44	11193	14204	14893
	E	38	38	44	2014	2379	2379
	F	38	38	44	4066	4798	4798
	G	46	38	44	6486	7679	7679
	H	46	39	46	10856	13147	16570
J2033_1	**A**	45	43	50	1845	2648	2775
	B	45	44	51	3690	5318	5574
	C	45	43	51	6165	8938	9262
	D	45	43	51	10620	15386	16041
	E	45	43	51	1845	2214	2464
	F	45	43	51	3690	4426	4426
	G	50	45	53	6150	7382	7745
	H	50	43	51	10450	12867	21597
J2033_2	**A**	37	38	43	1443	2915	2941
	B	37	38	43	2886	5756	5832
	C	37	38	43	4625	9178	9317
	D	37	38	43	7770	15493	15718
	E	37	39	43	1443	4095	4294
	F	37	38	43	2886	8237	8504
	G	37	38	43	4625	13052	13422
	H	37	38	43	7770	22075	22443
J2035_2	**A**	36	36	42	1548	2418	2418
	B	36	36	42	3204	4965	4965
	C	36	36	42	5220	8042	8042
	D	36	36	42	9000	13987	13987
	E	36	36	42	1548	2232	2232
	F	36	36	42	2304	4673	4673
	G	36	36	42	5220	7580	7580
	H	36	36	42	9000	13119	13119
J2036_1	**A**	35	32	37	1505	2035	2157
	B	35	32	37	2975	4225	4262
	C	35	36	41	4795	5634	5637
	D	35	35	42	8225	11762	11762
	E	35	36	41	1505	1768	1769
	F	35	36	41	2795	3496	3945
	G	35	35	41	4975	5635	5635
	H	35	35	41	8225	9672	9672

Instance Number	**Sub Instance**	Θ_{opt}	$\hat{\Theta}_{opt}$	Θ_{act}	$\boldsymbol{Z}_{opt}$	$\acute{Z}_{opt}$	$\boldsymbol{Z}_{act}$
J2036_2	**A**	32	29	33	1073	1446	1557
	B	32	29	33	1980	2647	2877
	C	32	29	33	3190	4171	4651
	D	32	29	33	5510	7253	8057
	E	35	28	33	980	1332	2187
	F	33	28	33	1848	2431	3021
	G	33	29	33	3003	3834	4924
	H	33	29	33	5247	6630	8598
J2037_1	**A**	31	31	36	2015	2373	2373
	B	33	33	38	3927	4617	4617
	C	33	34	38	6039	7093	7099
	D	33	34	38	9966	11716	11721
	E	31	31	36	2015	2372	2373
	F	33	33	38	3927	4601	4601
	G	33	33	38	6039	7071	7071
	H	33	33	38	9966	11658	11658