Solving Resource Constrained Project Scheduling Problems (RCPSP) with Remanufacturing

A Thesis

In

The Department

Of

Mechanical, Industrial and Aerospace Engineering (MIAE)

Presented in Partial Fulfilment of Requirements

For the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

January, 2019

Niloufar, Nasseri Bahrabad, 2019

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: Niloufar Nasseri Bahrabad

Entitled: Solving Resource Constrained Project Scheduling Problems (RCPSP) with

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair

Dr. Gerard Gouw

_____ Examiner

Dr. Ali Akgunduz

_____ Examiner

Dr. Anjali Awasthi

_____ Thesis Supervisor

Dr. Mingyuan Chen

Approved by _____

Chair of Department or Graduate Program Director

Dean,

Date _____

ABSTRACT

Scheduling is one of the crucial issues in the project planning phase. Completing the project in the desired duration with the available resources with minimum cost is a big challenge for project managers.

In the recent decades, several approaches have been proposed to deal with the resource constraints in scheduling. It can create a serious bottleneck and drastically change the flow of the activities. Moreover, resource constrains can change the project duration in crashing the project even if the activity (which creates the bottleneck) is not on the critical path.

To address this issue, a new approach for Resource Constrained Project Scheduling (RCPS) is proposed when the remanufacturing option for some activities is available in order to crash the project. In this research, first a mathematical model for RCPS is presented. Then, a new algorithm is proposed to shorten the project duration by activating remanufacturing line (if possible) or paying the crash cost. The proposed algorithm is implemented in MATLAB and some computational experiments have been done to demonstrate the effectiveness and sensitivity of the proposed procedures. The algorithm is also validated on a practical case study which is a manufacturing industry in the northern Ontario.

Acknowledgment

At the beginning, I want to thank God which was my comfort and assurance in all the difficulties.

The path toward this thesis completion is thanks in large part to the special people who supported, challenged and stuck with me along the way.

I would like to thank Dr. Mingyuan Chen, my supervisor for all his technical advices, encouragements, and financial supports. He helped me all the time when I needed, and he gave right direction into the compilation of the thesis. I could never have a better supervisor than him.

Besides, I also want to express my deepest thank to my dear friend, Dr. Iman Moazzen for all his numerous help and support during my study and being the best motivator all the time for me.

I am very grateful to all of my lab partners, Jaeir, Cesar, Omar, Zenan, Tiansheng, Dr. Bai and Rou for assisting me in several ways. Yasser Ghamary, thank you very much for all the experiences you shared with me. All the things that you mentioned were highly inspiring.

Nobody has been more important to me in the pursuit of my dreams than my family. I would like to thank my lovely parents, whose love, support, encouragement, and enthusiasm are always with me. Moreover my little sister who supported me emotionally far from home.

Table of Contents

List of Tables and Figures

CHAPTER 1: INTRODUCTION AND MOTIVATION

In this chapter, the general information regarding project scheduling is provided. Resource Constrained Project Scheduling Problem (RCPSP) is described. The challenges of the problem are discussed. The methodology to tackle the problem and the main contributions of the thesis are illustrated. Finally, an overview of the thesis outline is presented.

1.1 Project Scheduling

Despite the fact that finishing various projects in industrial units on time is considered as one of the effective factors in accelerating the socio-economic growth of countries, it is often observed that projects have been exploited with a high latency than estimated time and impose a high cost on the state budget. Investigations show that one of the effective factors in creating such defects in this field are the lack of planning or lack of proper using of scientific management methods.

Scheduling is one of the important issues in the project planning phase. The project scheduling is to determine the start time of each activity in project with respect to the constraints in order to achieve one or more specific goals.

It should be noted that project scheduling can be done in general or in detail. Project manager is virtually responsible for scheduling the project in general. In this type of scheduling, the deadlines are usually considered as important parts of the project and scheduling activities in detail is avoided. There are certain time periods which a set of activities or a stage of the project is completed. Usually, each contractor has an internal timetable for the project. It is used to designate the overall frame of the scheduling and details of the relevant activities. For project manager,

completion of the project before the deadline is pivotal. In recent years, extensive research has been done on project scheduling. Most of them assumed to have full access to all the information and facing deterministic problems. Scheduling, especially in project environments, has many applications. The first application of project scheduling is how to allocate resources to different activities throughout the planning horizon. The second application of it, is planning external activities such as material procurement, preventive maintenance, and order delivery to a domestic or foreign customer. All obligations of contractors to deliver materials and other project activities are done based on scheduling. Next, the main characteristics of the project scheduling are defined.

Definition of activities: This process involves identifying the necessary activities to achieve the objectives of the project which is determined by using the work breakdown experience and patterns (the list of activities of similar projects). Depending on the application of scheduling, activities can be defined in general or in details. The different factors are defined as follows:

1) Relationships between activities: This process involves identifying and defining precedent relationship between activities based on the: activity list, products descriptions, labor, environmental constraints, and also requirement dependencies. The activity network can be drawn by having a precedent relationship between activities.

2) Activity duration: This process involves estimating the duration of the project activities based on relevant information and records (time of activities in similar projects) and it is determined with opinions and estimates of experts. This feature is one of the main parameters of the project. Activity duration in some projects which are simple and repetitive can be deterministic, but in most projects, especially research projects or projects that are being carried out for the first time, are considered uncertain and stochastic.

In addition to the above three, there are other parameters that can play a decisive role in project scheduling; Including parameters related to resources such as access to resources and the usage rate for each activity in any time period. This information can also be determined according to the facilities and budget of the project, related records, and the opinions and estimates of experts.

1.1.1 Project Scheduling Techniques

In the following, we have a brief historical overview of the first project scheduling techniques. These techniques relatively simple to understand are the basis of all new and developing today's' methods in scheduling and project control.

i. Gantt chart

The first scientific considerations for achieving project planning and control were introduced by Henry Gantt and Frederick Taylor at the beginning of the 20th century. The two scientists used a graph for the project planning, whose horizontal axis represents the time factor and its vertical axis represents the activities for project implementation. These charts, later become famous as Gantt charts and bar charts. These are the simplest tool to show the start and end times of activities which are still used in many institutions and organizations as the only planning method. The main drawback of the Gantt chart is that the relationship between the dates of the activities in project and the order of precedence between them is not well understood. Therefore, if one or more delays occur, its effects cannot be understood on other activities and also on the completion date of the project easily. In the middle of the twentieth century, the existence of this failure led to the development and use of other planning methods.

ii. Critical path method

Critical path method (CPM) also called critical path analysis, was invented in 1957 as a tool of management, to improve, control and planning time from production to sales. The basis of this method is to find the longest pass of activity in the network. In CPM method, it is assumed that all activities can be carried out in their milestone and normal time. This method was initially devised to solve the cost-time tradeoff problem which project managers often encounter with. The goal of solving the cost-time tradeoff problem is to reach the earliest completion time of the project and to minimize the total cost. This is done by crashing the duration of some activities. The output of the cost-time tradeoff problem is to choose and define the activities should be shortened. In Crashing Problem with Critical Pas Method, the specific extra money is paid for each activity to reduce its duration. So the total duration of the project is reduced. First of all we need to find the critical path/paths in the network. Then among the critical activities find the cheapest one to crash. This activity have to be in the critical path. If it is not, the total duration will not change. If we have more than one critical path, we need to choose the activity which is common between critical paths or choose two different activities on each path and crash each of them if it's more economical. We continue this steps until the network reaches desired duration or non-feasible activity exists to be crashed.

iii. Project evaluation and review technique

In the late 1950s, the US Navy formed a team to design PERT technique. PERT is created as a statistical method to deal with the possible range of activity duration. The duration for each activity is divided to optimistic, pessimistic, and most likely term. Then, by applying these three times on a normal distribution curve, the expected duration of the activities are obtained. The success of the Polaris project in the 1960s has led PERT to be used as a planning tool in many companies.

Application of PERT and CPM methods has been quickly expanded in construction and industrial companies. Studies for developing these methods are still continuing, and so far, significant improvements have been made. These days in addition to time calculations and time/cost trade off problems, other issues such as resource leveling, human resources, and equipment can be solved by using these methods. In the years between 1957 and 1962, only about 1000 articles and magazines have been published in the United States. Each of them broadened the basic techniques in PERT and CPM.

iv. Graphical Evaluation & Review Technique (GERT)

GERT was invented in 1964 for projects with stochastics activities. First application of this method were in the projects related to the construction of a spacecraft. In the creation and development of this method, Dr. Paritzark had a significant influence, and wherever the GERT method is coming up, the works are done by him and his colleagues are reminded. As stated above, the basics of the most research which is related to the timing of the project are consist of simple basic techniques.

1.2 Resource Constrained Project Scheduling Problem Model

The issue of project scheduling can be divided into two categories in terms of the existence or absence of resource constraints:

Project Scheduling Problem (PSP) and Resource-Constrained Project Scheduling Problem (RCPSP). The limitations in the PSP model are just related to the pre-requisites relationships between activities; while, in the RCPSP model in addition to the prerequisite relationships between activities, there is also limited resources. Since in real projects, we usually face resource constraints, so PSP is concentrating more on the theoretical part and usually, the analysis and study

methods are first done on the PSP model and then expanded on RCPSP model. The RCPSP model is one of the most important and practical issues in project management. Since the emergence of this model in 1969, many studies have been done on it. In the following section, we will review the definitions and linear programming model in RCPSP.

1.2.1 Definition and Assumptions in RCPSP

In this part, we review the definition and assumption in RCPSP. The goal of solving RCPSP model is obtaining a feasible schedule so that the project is completed as soon as possible. Being a fusible answer means that all the constraints in the model are satisfied. In RCPSP two kinds of constraints have existed: the prerequisite relationships between activities and limited access to resources. The first constraint indicates that, in order to start an activity, all of its predecessor activities have already been completed. The second constraint also indicates that resources capacity which need for each activity are having limited access. Other assumptions of the basic model of RCPSP are as follows:

- The nature of the activities is single mode, it means that there is just one method exist for running activities. Activities can only be done with a constant combination of resource consumption and runtime.
- All resources are renewable.
- There is no discontinuity in activity scheduling, this means activities are scheduled continuously.
- All parameters of the model are considered deterministic

Some researchers have expanded RCPSP by made changes to any of the basic model assumptions. It is obvious that changing these assumptions leads to more complexity of the model. The most important changes in the researches are as follows:

1-Changes in assumptions about the nature of activities:
Single mode /multi modes of activity

As stated above, a single-mode activity can only be done in one form and with one scenario (in terms of the time it takes, the number of resources and costs). In a multi-state mode, you can perform activities with different combinations of runtime, cost, and amount of resources.

Preemptive/non-preemptive activity

If there is an assumption of non-preemptive activity, there should be no time disruption in the activity scheduling, and it should be scheduled continuously. While preemptive activity operations can be scheduled in a discrete manner.

Certainty / Uncertainty parameters

Information about each parameters of the project, such as activities, the level of access to resources, etc., can be uncertain.

2-Changes in assumptions about the nature of resources:
Renewable / non-renewable / dual resource

A Renewable resource is a resource whose level of access to it, is not dependent on its amount in any previous period of time. Therefore, the constraint associated with this type of resource is applied for each period of time, like machines and manpower. A non-renewable resource is a resource whose level of access to it at any given time depends on its amount in the previous period and will end due to consumption. Therefore, the constraint associated with this type of resource

applies to the entire planning horizon, like raw materials. A dual resource is a resource that has both of the above restrictions in combination. In this resource, the amount of access to the resource in the entire planning horizon is a constraint (non-renewable) and the amount of resource use per unit time has a limitation (renewable), such as a budget.

3-Changes in assumptions about the nature of precedence relationships**:**

- Precedence S-F / F-F / S-S / F-S

As stated, the precedence relations between the activities in the RCPSP model is that each activity can begin immediately after the end of its all predecessors activities. This kind of relationship is called a Finish-to-Start with zero time deference. The predecessors' relationship in CPM and PERT method is also F-S. In addition to F-S relationships, Start-to-Start (S-S), Finish-to-Finish (F-F) and Start-to-Finish (S-F) also can exist between activities. For example, an activity can begin after the start of all the predecessors' activities with the S-S relationship.

4-Changes in assumptions about the nature of the objective function:

- Regular /irregular objective

Regular objective functions are non- descending functions since the start of activities which aims to minimize it; this means that if the starting time of each activity delayed, there is no improvement in the objective function, Such as minimizing the completion time of the project. But postponing the starting time of activities in irregular objective functions may improve results, like maximizing the Net Present Value

- Single-objective / Multi-objective model

In a multi-objective model, more than one objective function is considered.

By combining the states described, various models are created. As mentioned, the one that most considered (Base model) is such that the nature of the activities are considered to be single mood, non-interruptible and deterministic. Resources are renewable and pre-requisites in F-S form with zero time difference. Often, this problem considers as a single objective, and the date of completion as an objective function; however, recent interest in multiple objective function and irregular functions has also increased.

1.2.2 Linear Programming Model in RCPSP

One of the main inputs of the RCPSP model is Project network. With a project network, a precedence relationship between activities is shown. Also, information regarding other parameters such as the duration of activities and the number of resources spent per activity can also be shown on the project network. Depending on how the activities are displayed through vectors or network nodes, each project can be displayed with Activity on Arrow (AOA) or Activity on Node (AON) network. However, Activity on Node has more application because of simplicity. In Figure 1.1, an example of AON network has been shown. The nodes and arcs between them, represent the activities and precedence relationships, respectively. The numbers above each node represent the durations and resource usage per activity. For example, the duration of activity 2 is equal to 4 units per unit of time and it use 2 units of resources in each unit of time.

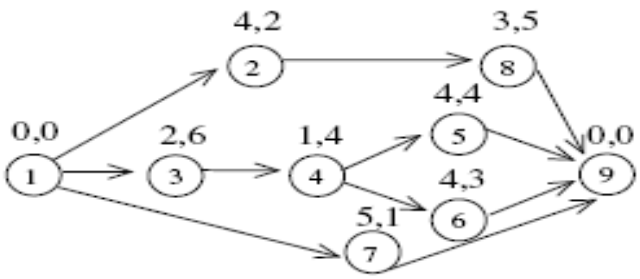

Figure 1. 1 Example of AON network

In RCPSP model, each project has N activities which two activities are dummies. The sources for most of the scheduling topics concentrate on sequencing and scheduling tasks on one machine. It should be noted that many of the production scheduling issues are the subset of the RCPSP model. For example, the job shop scheduling problem which is one of the major issues in production planning is a special case of the RCPSP model. Also, the resources in the job shop planning are the machines.

So far, to solve the RCPSP model many methods and algorithms have been developed. Some of the researchers have considered the exact methods but because of the NP-hard nature of the problem, most of them are using heuristic and metaheuristic methods.

1.3 Challenges and Motivations

Scheduling is one of the most challenging parts of the project planning phase. In this study, the resource constrained project scheduling problem by considering shortening the project makespan and adding a remanufacturing option to the algorithm is developed. Considering remanufacturing option not only can shorten the production duration in certain production systems, it is also a sustainable manufacturing practice. A lot of approaches have been proposed to crash network with the CPM method without resource limitation. However, crashing projects which have resource

constraint has more complexity. The CPM loses its efficiency when resource limitation is added to the network (Sonmez et al. 2015). A very simple example below shows that concept clearly. In this example, the first number on each activity defines the duration and the second one is the number of resources it needs. The maximum availability of the recourse is 5 units.

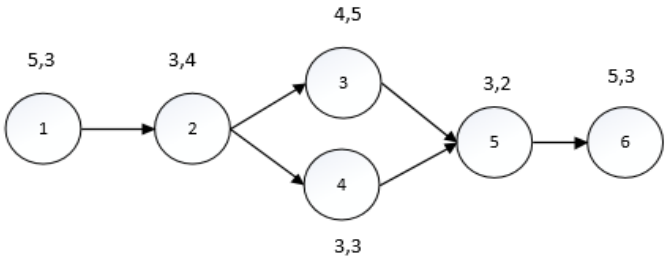

Figure 1.2 Network example for CPM method

As shown in Figure 1.2, the critical path includes activities 1, 2, 3, 5, 6 and the total duration is 20 time unit. First we assumed that the algorithm allocates resources to the activity with less duration. As a result, activity 3 should wait for activity 4 to finish, then it can start. So, if we choose activity 4 to crash, which is not in the critical path, it still changes the total duration of the project because it caused the activity 3 to start sooner.

To tackle this issue, a new approach is proposed. The new algorithm is presented which is used for crashing the resource constrained project scheduling model. Moreover, the remanufacturing option is added to the algorithm to crash the network easier. This is the first time that remanufacturing is used as an option for shortening the project.

In part one, the mathematical model for implementing RCPSP is presented. In part two, the crashing algorithm is implemented. Also, some numerical example has been done to test the effectiveness of the algorithm.

1.4 Contribution

This research proposes new mathematical model for Resource Constrained Project Scheduling. Moreover, the new algorithm is created to tackle the crashing problem with resource limitation. In the mathematical model, "off or on" concept is used. The new constraint for checking the precedence relationship between activities is developed based on this concept. Furthermore, by presenting the algorithm, crashing the resource constrained model are proposed. A new remanufacturing option is added to the algorithm to use for crashing the network. The objective of the model is to minimize the total cost of the project including crashing cast, remanufacturing cost, and penalty cost. The model is tested and validated using numerical examples with data generated by RanGen1 generator.

The algorithm include two parts. The first one solve the linear mathematical model to find out the best finish time of the project. After realizing the total makespan of the project in part one, in part two, the proposed algorithm tries to crash the project by minimizing the total cost.

1.5 Outline of the thesis

In this chapter, the general concepts and the history of the development of project scheduling techniques, the definitions and assumptions of the linear programming model of project scheduling with limited resources and various approaches to scheduling RCPSP are described. In the second chapter of this study, the literature review regarding RCPSP, Crashing problem, remanufacturing and disassembly approaches is presented. In the third chapter, the model under investigation of this research is defined and developed algorithm discussed step by step. In the fourth chapter, a

numerical example is explained in detail, to clarify all the steps of the algorithm. Also using some computational experiments for evaluating the performance of the proposed algorithm in different scenarios. At the end of the fourth chapter, the case study are analyzed and presented. Finally, in Chapter 5, the conclusions and suggestions for improving this algorithm are presented for future research.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

So far in the literature, to solve the RCPSP model many methods and algorithms have been developed. Some of the researchers have considered exact methods but because of the NP-hard nature of the problem, the methods used are mostly heuristic and metaheuristic. For crashing problem, most of the researchers use the CPM method, and some of them considered multi modes for uncertain situation.

2.2 Single-Project Scheduling Problem

In this section, we are going to consider different approaches with different objective functions in single project scheduling. Most of the researchers put their effort more on minimizing the makespan and cost. The other objectives like penalty minimization, net present value etc. are also considered.

2.2.1 Makespan Minimization

In the real world, the duration of a project can play a significant role. Therefore, a lot of researchers have tried to minimize the total duration of the project with different approaches. Mingozzi et al. (1998) has used 0-1 linear programming formulation to solve the classical RCPSP. The model is minimizing the makespan of the project. The approach concentrates on a critical path on the precedence graph to find the new lower bounds. The formulation includes three search algorithm which use new lower bounds to reach the optimal solution.

Tao and Dong (2017) have considered RCPSP with alternative activity chains to minimizing the total duration. By developing an integer linear program and also using AND-OR project network representation, a simulated annealing algorithm is proposed. By using new activity selection list on this simulation the large scale of problems is solved to optimality. The model is validated by using computational results.

The flexible project structure in RCPSP is presented by Kellenbrink and Helber (2015), they try to extend the general RCPSP model by using general model-endogenous decision on flexible project structure. By dividing activities into two optional and dependent, the power of flexible scheduling increases. They present a genetic algorithm to solve the problem and validate it using various numerical examples.

Bruni et al. (2016) considered uncertainty in activity duration in RCPSP and tried to reach to robust model. An assumption that the activity duration is controlled by interval uncertainty, is considered. Resource allocation decisions are taken in advance and the starting time of each activity can be adjusted to control the uncertainty. A general decomposition approach is led to solving robust RCPSP. For evaluating the approach, some benchmark examples have picked from PSPLIB and extensive computational study is conducted. The results of the model are shown the impact of the parameters on the algorithm performance.

Flexibility in resource profile with discrete time periods in RCPSP is introduced by Tritschler et al. (2017). They considered different time period for each activity and allocated resources to each of them to handle flexibility. A Hybrid Metaheuristic is embedded in the genetic algorithm. It uses the Flexible Resource Profile Parallel Schedule Generation Scheme (FSGS). The FSGS represents shorter makespans for the FRCPSP than a standard form of it.

In the real world, sometimes activities are executed in more than one mode. Therefore, an important extension of RCPSP in Multi-Mode Resource Constrained Project Scheduling Problem (MRCPSP) is introduced. When each activity has a possibility to operate in different activities' duration and different resource usage, the problem is called MRCPSP (Van Peteghem and Vanhoucke, 2014).

In Slowinski (1980) the multi-mode activities are proposed. In this model, resources are divided by three different categories of renewable resources (e.g. worker, machines), nonrenewable resources (e.g. materials) and doubly constrained resources (e.g. cash-flow per time-unit). Penalty cost not considered and also activities are allowed to have preemption. As an assumption of the model, the activities which are preempted, can start again later. Hence, Slowinski has constructed two linear programming approaches. First one is a one-stage approach that he just tried to solve the single-mode project scheduling with that. The second one is a two-stage approach which is connected two LP model together. It means that the outputs of the first LP model are the input data for the second one to find out the optimal solution.

Boctor (1993) has only considered renewable resources with minimizing the total project duration and non-preemption activities. He is made a comparison between 21 heuristics which are all developed by him. He tested all of them on 240 different examples. These examples divided into two groups of 50 and 100 activities. In 1996 by developing a new simulated algorithm which can solve both single and multi-mode activities, Boctor (1993) reached to the formulation that has ability to minimizing the project duration, net-present-value, and cost.

Drexl and Gruenewald (1993) approach is creating a mathematical model for multi-mode and also non-preemptive activities. Same as Slowinski, who is also considered all the three resources (renewable, nonrenewable and doubly-constrained). The goal feature of this model is to consider

all the changes regarding to job-specific resource profiles during time. In such a case, the usage rates of resources by each activity during the activating time in the project is not constant. An extension mathematical model is created for this problem. It is solved by stochastic scheduling method and reach to sub-optimality.

Hartmann (2001) proposed a genetic algorithm to minimize project duration. He also continued Slowinski's work and distinguished resources into renewable, nonrenewable and doubly constrained. He used two different local searches, one was checking the feasibility problem and the second one was checking the schedule found by GA to try and improve it. He believes that it is not possible to achieve good results by only using metaheuristic strategy. He also suggested that, paying enough attention to the problem specific representation is crucial for the success of the GA.

The eight agent-based algorithm and two types of agent-based systems (one is simple, and the other one is complex which is called enhanced agent-based) are developed by Knotts et al. (2007) to solve the makespan minimization problem. After testing all the eight priority rules, the result is shown that enhanced agents created the schedule with significantly shorter makespan.

Review of existing meta-heuristic for solving (MRCPSP) is presented by Van Peteghem and Vanhoucke (2013). Meta-heuristics are compared to each other. All the meta-heuristics are coded in a similar situation with the same stopping criteria to make a fair comparison and release all the computational results. Also, a new benchmark dataset is created. They suggested to the future researchers to compare their results to what they wrote in their paper.

Jozefowska et al. (2001) proposed a new simulated annealing approach to solve the MRCPSP problem with minimizing the total duration of a project. By considering SA without penalty function and SA with penalty function, two different versions of the simulations are discussed.

They applied three neighborhood (neighborhood shift, mode change and a combination of them) to both cases. All the examples generated in ProGen and a vast computational experiment is to examine. From the results, it is stated that by adding penalty function, the simulated annealing approach performs better.

2.2.2 Cost Minimization

Minimizing the cost of the project is always an important matter for all the managers. One of the most expensive aspects of project scheduling is renewable resources and it also significantly impacts on the cost of the project. Demeulemeester (1995) introduced the Resource Availability Cost Problem (RACP). The objective function is to minimize the single project cost of allocating recourses to project. RACP is very similar to RCPSP in basic and has a few discrepancies in the objective function and some constants.

Salewski et al. (1997) introduced Mode Identity Resource Constrained Project Scheduling Problem (MIRCPSP) with cost minimization and non-preemptive activities. Renewable and nonrenewable resources are defined. Each job set consists of some subsets, and all the activities which are being in the same subgroup should be executed in the same mode. They proposed a solution methodology as a tailored parallel randomized (RAMEZ) approach. This approach can be implemented in both static and dynamic priority rules.

Debels et al. (2004) generated a new meta-heuristic by a combination of three methods which can provide near-optimal solutions for large examples. First is elements from scattering search, second is a generic population-based evolutionary search method, and the last is one of the previously introduced heuristic methods. After doing a vast amount of computational

experiments on the standard benchmark datasets, it is realized that the new algorithm outperforms all the heuristics in the literature.

Shan (2015) proposed the demand-driven problem scheduling in aircraft assembly system and also resource constraint project scheduling (RCPSP). The genetic algorithm is used to solve the problem. Some innovation is considered in the genetic algorithm (GA) (i) for implementing the process, two crossovers and three mutations are used. (ii) Its suitableness function is demand-driven. Finding the optimal combination of working time, operators and space are the most important objectives in RCPSP for aircraft assembly. For validating the algorithm, by collecting the real demand's data, two encoding approaches are tested.

After reviewing Resource Constraint Project Scheduling, in the rest of the literature Project-Crashing Problem, Remanufacturing and Hybrid system with manufacturing and remanufacturing option are also taken into account regarding the goal of this thesis considers the following topics as well.

2.3 Project-Crashing Problem

In this challenging world, the makespan of the project is completely pivotal for both managers and stakeholders. Hence, a lot of research is done to find the best method for shortening the total project by spending less money. The project crashing problem is reviewed under two total categories. Some of the researchers have examined the problem under deterministic circumstances and others have considered uncertainty in the problem. Below some of the accomplished studies are discussed.

2.3.1 Deterministic Methodology (CPM method)

Since the critical path is a network optimization problem, for shortening the project by CPM at first, we need to identify the critical path and critical activities. In the next step, we use one of the following methods:

- Pruning critical path activities
- Fast-tracking" (performing more activities in parallel) ”
- Crashing the critical path" (shortening the durations of critical path activities by adding resources) ”

A mathematical model for crashing the Multi-Period Multi-Product (MPMP) problem is proposed by Feylizadeh et al. (2008). The objective function minimizes the total cost of crashing in each operation to decrease the makespan of the project. The capacity of machines is assumed constant. A multi-stage mathematical model is used to crash the production process.

Roemer and Ahmadi (2004) presented a model and used two different methods to solve it (makespan minimization). One is crashing, and the other one is overlapping the activities. The objective of their research is to find optimized overlapping/crashing policies. The method which computes the adequate borderland of time-cost trade-off is proposed. By illustration of a two-stage example, the consequences of the important parameter and the robustness regarding their assessment are disguised.

Particle swarm optimization (PSO) algorithm for project-crashing problem is developed and tested by Yang (2006). A new model is developed which is capable of dealing with both time-cost linear and non-linear problems. It helps managers to implement what-if analyses for deciding about budget allocation and also meet deadlines and delay penalty of projects.

Pulat and J. Horn (1996) proposed a multiple objective linear programming (MOLP) model for time-resource trade-off problem to construct the best scheduling for allocating two resources. For each activity, the normal duration, the maximum allowable crash time and resource cost per time unit are considered. As a solution methodology, the enumerative and interactive algorithms which are exploited to Geoffion's $\rho(\lambda)$ approach and the time-cost trade-off technics are used. The maximum flow procedure used to determine the critical subnetwork and crashing the activities with the lowest cost.

Mohanty et al. (2011) used simulation influenced by a method which was implemented on an electronic spreadsheet. The simulation is used to identify activity modes for project crashing. Also, it is capable of solving discrete time-cost trade-off problem. The resources are considered unlimited, and the time-cost function is continuous. The simulation shows promising results which are close to optimal. The spreadsheet simulation approach is very user-friendly and helps a user to implement what-if analysis.

2.3.2 Uncertain Methodology (PERT)

Program Evaluation and Review Technique (PERT) is widely used in most of the research and improvement projects which are faced with uncertainty.

Coskun (1984) is the first researcher who addressed the problem of crashing by considering stochastic conditions. He used chance constrained linear programming (CCLP) to formulate the problem. This method is aimed to convert stochastic mathematical formulation into the similar deterministic formulation. He also assumed the activity duration is followed the normal distribution instead of beta distribution. Also the mean and standard deviation are known. Coskun

concluded that "While the solution of the CLLP formulations of the optimal PERT compression problem provides a wealth of information with significant managerial implications, the computational efforts necessary to solve the CCLP are no greater than those necessary to solve the deterministic compression problem."

Gocken (2012) concerned the project crashing analysis for a multi-objective problem. The uncertainty is modeled via fuzzy set theory. Direct solution approach is created to solve fuzzy theory. The Tabu search metaheuristic algorithm and choosing fuzzy numbers with the ranking method are used in the direct solution method.

Liu (2003) suggested the solution based on Yager (1981)'s method which is used fuzzy formulation, but here it changed to a crisp linear problem. The theory is implemented to the critical path and the project crashing problems. The fuzzy completion time of the project is evaluated with most likely value and also the probable completion time in pre-defined range. Also, crash cost assumed to be crisp values. As a result of this development, more information regarding time and cost factor of the project provided for a manager to make the decisions. A. Haga and O'keefe (2001) proposed the simulation model which is aimed to find the best activity for crashing by considering the factor like cost, bottleneck, and duration of activity, etc. in the project. The simulation result is finding out the best crashing strategy for a PERT network and also minimizing the penalty cost as a function for late project completion deadline. The author mentioned that this research was investigated from Haga (1998) which is done for his Doctoral dissertation but never published.

Aghaie and Mokhtari (2009) presented the new approach for solving the PERT crash-project problem based on ant colony optimization (ACO) metaheuristic and Monte Carlo (MC) simulation technique. The objective function of this approach is to boost the probability of project completion

optimally in a pre-stated deadline. The model first finding the critical path by using MC simulation technic. Afterward, ACO is implemented for path selection by solving the nonlinear integer mathematical programming.

2.4 Remanufacturing

In today's world, remanufacturing is becoming the priority for managers to eliminate environmental pollution and also decrease the total cost of the projects. Hence, in recent decades more researcher pays attention to this topic and try to propose different strategies and methodologies. Here some of the most relevant to this thesis are discussed.

Hatcher et al. (2011) discussed a vast literature review in design for remanufacturing and suggested the gaps to researchers for future topics.

In the same year, Lage Junior and Filho are written another literature review. The main focus of it is on the production planning and control (PPC). Seventy-six papers are checked out, and categorized.it is mentioned that none of the examined papers are considered all the complexities of remanufacturing at the same time and more practical research is needed in this area.

Another literature review is done by D. Morgan and J. Gagnon (2013). The main focus in this literature is scheduling of remanufacturing operations. The approaches which are created to overcome product deterioration as an additional complexity are discussed.

Lee et al. (2001) reviewed the scheduling in disassembly systems. The sequences of disassembly and the different design types of products, redesign issues are considered. The fundamental

suggested for future research topic is presented as making solidarity between disassembly planning and scheduling problems.

Ferrer and Whybark (2001) considered some real firms to find out the best material planning for remanufacturing. The remanufacturing steps are as follows: first of all using end-of-life component, then disassemble them, find out the useable items from disassembly parts and assemble them again. The uncertainty is applied to the demand rate for remanufacturing products, the rate of good parts in disassembly items and also the supply of end-of-life goods.

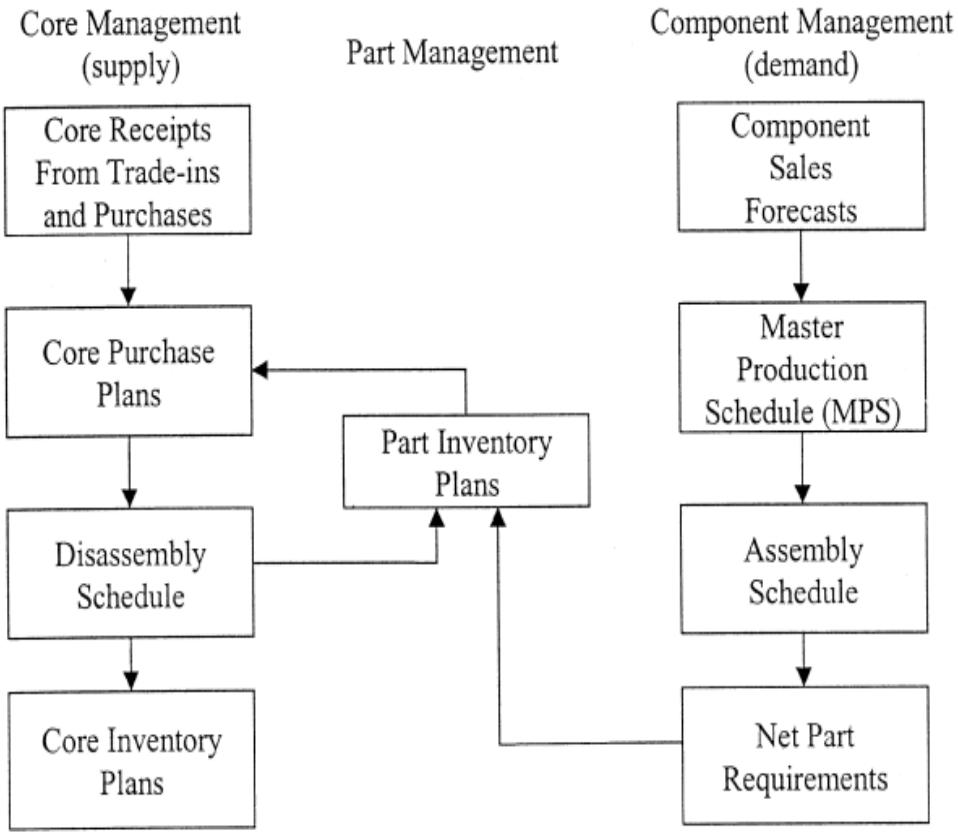

Figure 2.1 Information diagram for remanufacturing Ferrer and Whybark (2001)

Zhou and Tang (2012) analyzing a multi-period stochastic dynamic program for managing a remanufacturing by considering uncertainty on demand and returned rate of serviceable products. The objective of this research is finding an optimal ordering/remanufacturing policy. As the problem is complex, three simple heuristics are developed.

From the numerical example, one of the best out of three heuristics is introduced.

Nakashima et al. (2007) presented remanufacturing system with stochastic demand. Markov decision process is considered for system formulation. Remanufacturing is applied in two different types of inventories, one is the available inventory in a factory, and the other is the virtual inventory which is going to be used by customers.

Li et al. (2010) proposed the stochastic dynamic programming model for analyzing the production planning. The objective of this model is to minimize the total cost of the project by recognizing the optimal amount to be remanufactured. The rate of return amount and also the demand are considered uncertain. The policy iteration is used to find out the least costly plan of a remanufacturing system. Finally, the numerical example is used to validate the feasibility of the model and show how it should be implemented in the production planning.

The production planning and control activities for seven difficult characteristics are identified and discussed by Daniel and Guide Jr (1999). Also suggested the research opportunities for each of them.

2.5 Hybrid System

In recent years, merging manufacturing and remanufacturing is becoming an important research area. Researchers try to create models and methods to help the companies figure out the best combination of manufacturing and remanufacturing for each of them. S. Zanoni et al. (2011) one of them. The paper introduced a shift PULL inventory policy and compared it with PULL, DUAL and Separate PULL control policies. Simulation is used to solve the problem. In the end, two main suggestions are made. 1) In the situation that manufacturing is significantly taking longer than remanufacturing, it is better to separate PULL and DUAL for better results.2) When the difference between the lead time of manufacturing and remanufacturing is noticeable, the shifted PULL has a better performance.

Laan et al. (1999) compared a proposed model to the traditional system without remanufacturing to give the manager a better view of inventory decisions. The assumptions are as follows: 1) Both output of manufacturing and remanufacturing can use to meet the customer demands. 2) A single component is used. 3) The PUSH and PULL control strategy is considered with remanufacturing option.

Fazle Baki et al. (2013) developed the mixed-integer model formulation, aims to minimize the cost of production plan with dynamic lot sizing and remanufacturing. With this approach, every optimal solution has the ability to break up into blocks with distinct patterns and identified that which blocks occurred in manufacturing and which one in remanufacturing.

CHAPTER 3: PROBLEM DEFINITION AND FORMULATION

3.1 Introduction

In this chapter, resource constrained scheduling problem with crashing the project is studied. The mathematical model is developed, and the definition of objective function and constraints are described. The model presented project by activity on node network (AON). Each project starts and finishes with dummy activities. The resource usage and duration for dummy activities are zero. We assumed all the information regarding resource usage of each activity and also the duration of them is known and deterministic.

3.2 Definitions

This model insists on following the conditions which help to be close to the real situation. Hence, the practical assumptions are considered. Below some of the definitions are explained.

i. Resources

The renewable and non-renewable resources are considered in this study. A renewable resource can use for more than one time. In other word, it is not consumed and finished during the project. It means when one activity finishes, the renewable resources of it are ready for the next activity. For example machines and workers are considered as renewable resources. A non-renewable resource is a resource which able to use just for one time. It means when you allocate non-renewable resource to each activity, it is consumed after processing, and you cannot allocate it again to another activity. The non-renewable resource is like raw material. In this model at time

zero, the usage amount of both renewable and non-renewable material for each activity and the availability for each of resources are known and deterministic untill the end of the project. At the beginning of the time horizon, the resources are allocated to all the activities in the project. For starting each activity, all the required resources should be available. The precedence relation in all the project is finish-to-star so each activity can start if and only if all the predecessors are done before.

ii. Bonus and penalty

To be more close to the real world and to have a realistic model, the bonus and penalty have included in the algorithm. The bonus is the money that a customer pays to the company if the project is finished earlier than the deadline. The bonus amount for each project is different. We assumed it is calculated by multiplication of the bonus amount for each day and the number of the days which project finished earlier than the due date. The penalty is the amount of money which company pays to customer for each day delay in the project. All the company tries to avoid having a delay in their project and meet the deadline as possible as they can.

iii. Setup cost

Setup cost is the amount of money should spend for set up a specific equipment and get it ready for production processes like activating machines, production line or assembly line. This is the cost incurred for changing tools, labor cost of setting up the equipment and etc.

iv. Normal time

Normal time for each activity is the time it needs to start, process and finish. Also, it is called activity duration.

v. Normal cast:

It is the lowest possible direct cost which spends on each activity for the regular requirement in an ordinary condition like resources, manpower ad etc. to complete an activity.

vi. Crash cost:

In many projects, it is desirable to reduce the total project time, even if it caused to increase in cost. In such a case, the crash cost is defined as the extra cost for accelerating the completion of projects. Usually, by adding more resources to one or more critical activities, the completion process is going to be faster. The cost is unique for each activity. It is calculated by manipulation of the crash cost for each activity and the time/unit we want to crash the activity.

vii. Crash time:

The number of time-unit which we allow to shorten the activity is called crash time. On the other word, it is the shortest possible time to complete each activity. Also, we have another important definition as a total allowable crash time which shows the difference between normal time and crash time.

viii. End-of-life product (EOL):

When an item, product, equipment or machine riches to the end of its useful life cycle, it is called end-of-life. It is the last stage of a product's lifecycle which is started with design, improvement, released and used.

3.3 Problem formulation

In the work by Ulman (1975), it is shown that scheduling is categorized as an NP-Complete problem. Hence, the problem that is studied here is also NP-complete with more complexity than

a classic scheduling problem. A new algorithm to solve the problem is developed. The approach consists of two stages. With a lot of instances, each in a different situation, the model is validated, and the algorithm is checked. MATLAB is used as the software to solve the problem. There are N activities named as $[a_1, a_2 \ldots\ldots , a_N]$. The precedence relationships among activities are defined in P Set, where each row is dedicated to a pair of dependent activities. For example, the flow of activities for the following P:

$$P = \begin{bmatrix} a_1 & a_2 \\ a_1 & a_3 \\ a_2 & a_4 \\ a_3 & a_4 \end{bmatrix}$$

Is shown in Figure 3.1:

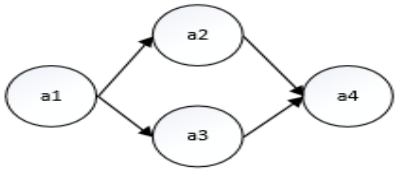

Figure 3.1 Simple network example

The normal duration for activities, minimum duration after crashing, normal cost and the crash cost are defined as D, C, NC, CC respectively:

$$D= [D_1, D_2 \ldots\ldots , D_N]$$

$$C= [C_1, C_2 \ldots\ldots , C_N]$$

$$NC= [NC_1, NC_2 \ldots\ldots , NC_N]$$

$$CC= [CC_1, CC_2 \ldots\ldots , CC_N]$$

Here we presented the algorithm which is able to solve the resource constrained project scheduling with adding crashing problem and remanufacturing option. The algorithm is divided in two different parts. In the first part, the resource constrained project scheduling with two different types of renewable resources is solved. In the second part, the crashing problem is implemented by having a remanufacturing option for some activities.

3.3.1 Part 1: Resource Constrained Allocation

The objective is to minimize the total project duration and allocated the limited resources to all the activities in an optimized way. To formulate this problem, many researchers considered the starting date for each activity. This makes it more difficult to integrate the resource allocation constraint into the optimization problem.

In this regard, we proposed a new optimization problem using activity status at each time unit (off, on) and find the solution using an iterative approach. The algorithm is summarized below:

Initialization: Set the current duration $[d_1, d_2 \ldots\ldots , d_N]$ to the normal duration.

Solve the following optimization problem.

Sets:

T - Set of time periods, t= 1...T

A - Set of all activities A= $a_1, a_2, \ldots\ a_n$

P - Set of all precedence relationships

Parameters:

r_n^{ren1}- *The required renewable resources type one in activity n*

r_n^{ren2}- *The required renewable resources type two in activity n*

R^{ren1}- *The maximum availability of renewable resources type one*

R^{ren2}- *The maximum availability of renewable resources type two*

Decision variables:

$$a_n\,(i)\begin{cases}1 & if\ the\ activity\ n\ is\ active\ in\ time\ unit\ i\\ 0 & otherwise\end{cases}$$

Objective function:

$$\text{Minimize } (fx)$$

$$f = [0_{1*(N-1)T}, 1,2,3, \dots, T] \tag{1}$$

$$x = [a_1^{(1)}, a_1^{(2)}, \dots\dots, a_1^{(T)};\ a_2^{(1)}, a_2^{(2)}, \dots\dots, a_2^{(T)}; \dots;\ a_N^{(1)}, a_N^{(2)}, \dots\dots, a_N^{(T)}]^T$$

Contraints:

S.t:

$$a_n\,(i) + a_n\,(j) \le 1,\ \ n= 1, 2, 3, 4 \dots N, i = 1, 2, 3 \dots T\text{-}d_n\ ,\ \forall\ n, I \mid j = i + d_n, \dots, T \tag{2}$$

$$\begin{cases} a_n\,(j)j - \sum_{k=1}^{j-1} a_m(k) \le j - d_m & for\ j \ge d_m \\ a_n\,(j) = 0 & for\ \ j = 1{:}\ d_m - 1 \end{cases},\ \ \forall\ m, n\ \in\ p \tag{3}$$

$$\sum_{n=1}^{N} a_n(i) r_n^{ren1} \le R^{ren1} \qquad for\ i = 1, 2, \dots, T \tag{4}$$

$$\sum_{n=1}^{N} a_n(i) r_n^{ren2} \leq R^{ren2} \qquad for\ i = 1, 2, \dots, T \qquad (5)$$

$$\sum_{i=1}^{T} a_n(i) = d_n \qquad \forall\ n = 1, 2, 3, \dots, N \qquad (6)$$

The objective function of the problem modeled in Eq(1) is the minimization of total project makespan. It considers the last activity of the project and tries to minimize it. This ensures the fastest possible completion time for the last activity (equivalent to the shortest project duration). The zero weight is given to all the activities except the last one. The cost for the last activity is increasing as time progresses. So the dedicated weight is increased day by day. To satisfy the goal of the objective function (cost minimization), the model tries to finish the last activity as soon as possible. After convergence, the last activity time unit of a_N will set the current project duration. Constraint in Eq(2) ensures that the activities are non-preemptive, it wants to make sure that when the activity starts, it will continue until its duration ends. In constraint in Eq(3) it is guaranteed that the precedence relations between activities are executed in order. Inequality in Eq(4) and Eq(5) emphasizes that the total resources of renewable1 and renewable2 which are allocated to all the active activities do not exceed the total availability of them. Inequality in Eq(6) indicates that the total active days for each activity is being equal to the total duration of it.

3.3.2 Part 2: Project Crashing with Resource Constrained and Remanufacturing Option

As mentioned before, having a remanufacturing option for shortening the project is never considered by previous researchers. In this part, the remanufacturing option is added to the algorithm. In each project, we assume some activities have access to remanufacturing line if desired. The associated cost to use the remanufacturing option for an activity is as follows:

r_n^{non} = number of non-renewable resources that activity n need

C_n^{rem}= Setup cost + modification cost *r_n^{non}

It was assumed when the n^{th} activity uses the remanufacturing option (if allowed), r_n^{ren2} becomes zero. Furthermore, the dependency on all the previous activities (except the start activity) will be removed. Also, in some cases (when the eliminated activity is just being the predecessor of the remanufacture ones) all the renewable resources of the previous activities will be available to allocate again. To illustrate this, let's consider a project with the activities flow as shown in Figure 3.2.

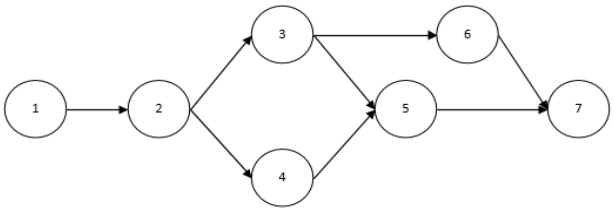

Figure 3.2 Regular network

For simplicity let's assume only 3^{rd} and 6^{rd}activities have the option to choose the remanufacturing line. The activities flow for the case that the 3^{rd} activity just uses the option is shown in Figure 3.3. Note, the second activity cannot be totally bypassed as the fourth activity depends on it.

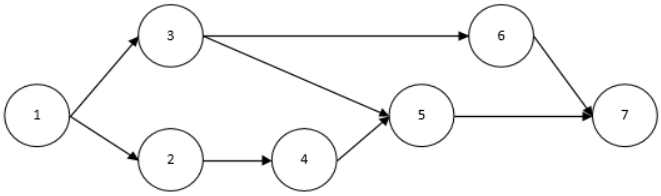

Figure 3.3 3rd activity is remanufactured

The activities flow for the case that the 6^{th} activity only uses the remanufacturing option is shown in Figure 3.4.

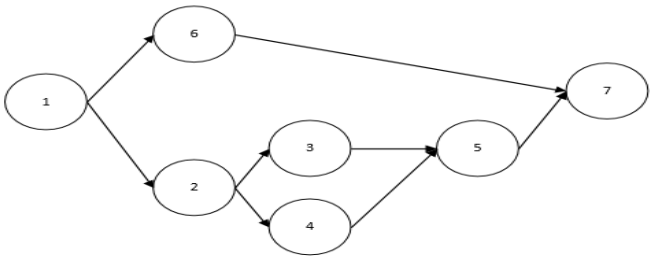

Figure 3.4 6th activity is remanufactured

The activities flow for the case that both activities use the remanufacturing option is shown in Figure 3.5.

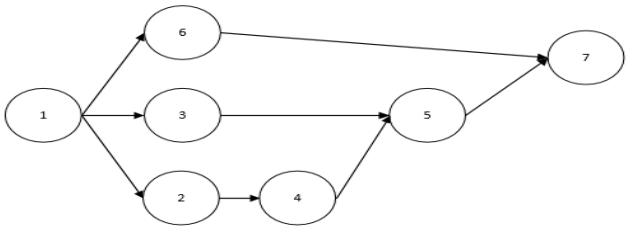

Figure 3.5 Activities 3 and 6 are remanufactured

For solving the part 2 the steps illustrated bellow is needed to pass:

Let's assume a^{rem} include activities that have access to remanufacturing option.

Step 1) find all possible permutations for remanufacturing including no remanufacturing, remanufacturing of any individual activity in a^{rem}, remanufacturing of any pair of activities in a^{rem} and so on.

Step 2) for each possible option, create the dependency set, i.e. P.

Step 3) for each set, reduce d_n (current duration of the n^{th} activity) by one unit (if feasible) and solve the optimization problem in step 1 for n=1, 2…, N. For each case, record the total project duration after crashing and the crashing cost for that activity, i.e. $\frac{CC_n - NC_n}{d_n - C_n}$.

In other word, the algorithm reduces one unit from normal duration of all the activities one by one, and back to part 1 to solve the optimization problem and check if the project duration has been reduced after crashing an activity or not.

Step4) if not, for feasible reduction scenarios, add the corresponding cost by a large penalty. This enforces the solver not to select that activity for crashing unless no other option is available.

Step 5) if it has been reduced, for feasible reduction scenarios, divide the crashing cost for that activity by the number of reduction. This reward (cost reduction) acts as an incentive to select an activity for crashing when activity crashing by one unit leads to more than one unit of reduction in project duration.

Step 6) select the activity that has the minimum cost as the crashing winner. And update the current project duration.

Step 7) if the total duration is greater than desired and further crashing is possible, go to step 3.

Step 8) for each option found in step 3, the cost will be calculated as follows:

$$\text{Cost (option}_i) = \begin{pmatrix} total\ crashing\ cost \\ for\ all\ the\ crashed \\ activities \end{pmatrix} + \begin{pmatrix} total\ remanufacturing\ cost \\ for\ all\ activities\ that \\ used\ the\ option \end{pmatrix} + \begin{pmatrix} total\ delay \\ penalty \end{pmatrix}$$

Where:

$$\text{Total delay penalty} = \left(\begin{pmatrix} finall\ duration \\ for\ option_i \end{pmatrix} - \begin{pmatrix} desired \\ duration \end{pmatrix} \right) * \begin{pmatrix} penalty \\ per\ time \\ unit \end{pmatrix}$$

Note that for the case which a remanufacturing option is active for the n^{th} activity, r_n^{ren1} is set to zero.

Total delay penalty will be non-zero when further crashing in the i^{th} option is not feasible and the desired duration has been achieved.

CHAPTER 4: EXAMPLE PROBLEMS AND COMPUTATION

In this chapter, the developed algorithm, the data used in the algorithm and the results using optimization are shown and a comparison is made.

4.1 Data description:

The popular generator RanGen1 is used for generating the data used for this study which was introduced by Demeulemeester et al. (2003). OS is defined as Order Strength which means the number of precedence relations which includes only the transitive ones divided by the theoretical maximum number of precedence relations in the network. So the OS is calculated as $(n-1)/2$, which n is the number of non-dummy activities in the network. The OS is shown the complexity level of the network. By increasing the OS number, the complexity of the model is increased too. RanGen1 starts the generation process based on the number of activities and the OS as an input. Project network is considered for calculating the time window for each activity. The rest of RanGen1 parameters are described in Table 4.1.

Table 4.1 Information of RanGen1 parameters

The values used for ProGen parameters	
Number of non-dummy activities	i-2
Number of dummy start/finish activities	2
Number of successors/predecessors per activity	[1,3]/[1,3]
Order Strength of network	0.6
Number of renewable/non-renewable resources	2/1
Duration of each activity	[4,10]
Renewble1/Renewble2/non-renewable resources demand per period	[1,5]/[3,7]/[3,7]
Renewable resource factor/strength	(0.4/0.1)
Non-Renewable resource factor/strength	(0.9/0.6)

The example is generated with i+2 activities, including dummy activities. Each activity use all type of considered resources. The number of renewable and non-renewable resources is fixed and known.

4.2 Worked Example

In this part, the details of the tested example instance for the algorithm is elaborated. All the steps and approach taken in each section are studied, and the results are explained. The simple network

is chosen for the better understanding of the algorithm steps. In the next part, more complex examples will be explained. The diagram for this instance is shown in Figure 4.1.

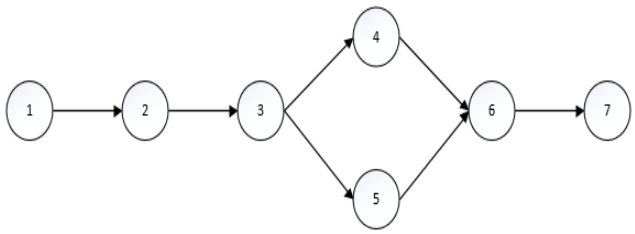

Figure 4.1 Work example network

As it is illustrated, the project has 7 activity (two dummy activities). Two of the non-dummy activity can use the end-of-life items from remanufacturing line. The activation cost for remanufacturing line is $100, and the modification cost for each needed non-renewable resource is $10. There are two types of renewable resources (workers and machines) for each activity. Also for calculating the remanufacturing cost, we have to know the usage amount of non-renewable resources for the activities. The maximum availability of renewable resources is ten and seven respectively. To make it nearer to a real industry situation we add penalty cost too. It is equal to (max crash cost per time unit)*2. By considering the mentioned equation, in this example, the penalty cost is $154 per each day delay. The rest of the information is illustrated in the Table 4.2.

Table 4.2 Network information for work example

Activity	Normal duration	Crash duration	Renewable 1(worker)	Renewable 2(machine)	Non-renewable	Normal cost	Crash cost
1	0	0	0	0	0	0	0
2	7	5	3	6	4	96	286
3	6	3	2	5	4	92	268
4	6	4	5	3	3	77	336
5	8	4	3	5	4	65	312
6	9	7	1	6	4	83	305
7	0	0	0	0	0	0	0

To explain the process of the algorithm step by step in both two part we have:

At first, in part one the optimization problem is solved and satisfied all the constraint in the mathematical model. In this optimization, the best resource allocation with the available resources is decided with considering the minimization of the total project duration.

After running the model in MATLAB, the resources are allocated to the activities as below. In lack of resources, the model allocated recourses to the activity which has a greater duration.

The total project duration is 36 days. As the resource usage for dummy activities are zero, they are not considered in resource allocation.

Table 4.3 Resource allocation in Case 1 work example

Day / Activity	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
2	*	*	*	*	*	*	*																													
3								*	*	*	*	*	*																							
4																						*	*	*	*	*	*									
5														*	*	*	*	*	*	*	*															
6																												*	*	*	*	*	*	*	*	*

In part two, after realizing the total project duration (36 days), the algorithm analyzes all the possible option for crashing the project to reach to desired duration (15 days) in step 2. For the sake of visualization, a large discrepancy is selected between total project duration and desired duration. In this example, as there are two possible remanufacturing options (activity 4 and 5), we have four different cases to consider: 1) no remanufacturing, 2) activity 4 is remanufactured, 3) activity 5 is remanufactured, 4) both activities 4 and 5 are remanufactured. In each case the total crashing cost and the total project reduction is different.

In step 3, the precedence relationship between activities is influenced by each different case. In other word, the algorithm creates a different P set by considering each case. Here we have:

Table 4.4 Precedence relationship in work example

cases	**Dependencies**
Without Remanufacturing	{1,2;2,3;3,4;3,5;4,6;5,6; 6,7}
Activity 4 Remanufactured	{1,2;2,3;**2,4**;3,5;4,6;5,6; 6,7}
Activity 5 Remanufactured	{1,2;2,3;**2,5**;3,4; 4,6;5,6; 6,7}
Activity 4&5 Remanufactured	{1,2;**2,4**;**2,5**;4,6;5,6; 6,7}

After having all the information regarding each case, in step 4 the algorithm starts to do crashing the network by reducing one unit of the normal duration for each activity in each case founded in

step 2. From here, the details of each case are discussed separately, because each of them has a different processing steps. Then the compression between them is made.

Case 1:

As mentioned before, in case one no remanufacturing option will be available for activities. And the network is the same as the one shown in Figure 4.1. So the only way to reach the desired duration is paying the crash cost and shortening the project. Here we have:

Table 4.5 Information of each iteration in Case 1

Iteration	Activity	Normal Duration	Final Duration	Crash Cost
1,2,3	3	6	3	60
4,5,6,7	5	8	4	142
8,9	6	9	7	91
10,11	4	6	4	115
12,13	2	7	5	154

The maximum allowable time, to crash all the activities is 13 days. It costs $562. The makespan of the project now is 23 days. So we still need eight more days, but it is impossible to crash more. Because all the activities crashed by their total allowable crash time. Hence, we have to pay $1232 as a penalty cost for the rest eight days. The total cost for this case is $1794.

Case 2:

In this case, activity 4 is using remanufacture option. In this situation, three changes happened and helped to speed up the network.

1) The first change happens in the precedence relationship. Here, activity 4 can start immediately after finishing activity 2, and there is no need to wait for activity 3 to complete.

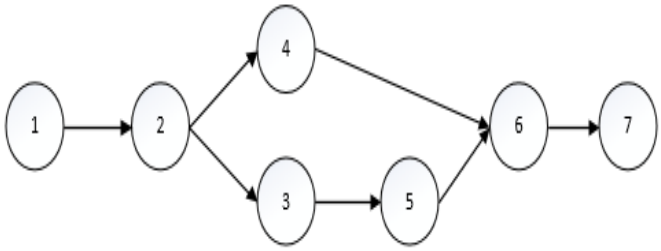

Figure 4.2 Network in Case 2

2) The usage of renewable type 2 (machines) for this activity is changed to zero. So it will be free to use for other activities. Here the total availability of renewable2 is changed from 7 to 10. The resource allocation for new network is:

Table 4.6 Resource allocation in Case 2

Day / Activity	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
2	*	*	*	*	*	*	*																							
3								*	*	*	*	*	*																	
4								*	*	*	*	*	*																	
5														*	*	*	*	*	*	*	*									
6																						*	*	*	*	*	*	*	*	*

As shown in Table 4.6, after remanufacturing activity 4, the total project duration decreased to 30 days.

3) The remanufacturing cost added to total cost. It is calculated below:

100$ + (3)*10= $130

After implementing all the changes related to remanufacturing activity 4, we still did not reach to 15 days. So the crashing is used. The detail of it is in the Table 4.7.

Table 4.7 Information of each iteration in Case 2

Iteration	Activity	Normal Duration	Final Duration	Crash Cost
1,2,3	3	6	3	60
4,5,6,7	5	8	4	142
8,9	6	9	7	91
10,11	2	7	5	154

The network crashed 11 days. It costs $447. The makespan of the project now is 19 days. So we still need four more days to reach the desired reduction. Although activity 3 is not crashed, it is impossible to crash the network more. It means the algorithm cheeked and realized that if activity 3 is crashed, the total duration is not changed. So the algorithm adds extra amount ((max crash cost per time unit)*2) to the crashing cost of activity 3, to make sure it is not chosen as a crash winner. Hence, we have to pay $616 as a penalty cost for the rest 4 days. The total cost for this case is $1193.

Case 3:

In this case, activity 5 is using remanufacturing option. All the changes mentioned above are happened here as well.

1) Precedence relationship is changed and activity 5 does not need to wait for activity 3 to be completed anymore. The new network is :

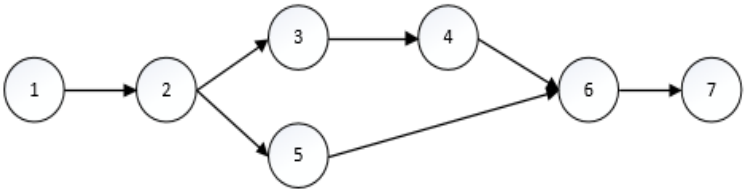

Figure 4.3 Network diagram in Case 3

2) The usage of renewable type 2 (machines) for this activity is changed to zero. So it will be free to use for other activities. Here the total availability of renewable2 is changed from 7 to 12. The resource allocation for the new network is:

Table 4.8 Resource allocation in Case 3

Day / Activity	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
2	*	*	*	*	*	*	*																					
3								*	*	*	*	*	*															
4														*	*	*	*	*	*									
5								*	*	*	*	*	*	*	*													
6																				*	*	*	*	*	*	*	*	*

As shown in Table 4.8, after remanufacturing activity 5, activity 3 and activity 5 can start at the same time. Moreover, because we still have available resources, the activity 4 is start immediately after 3. So the total project duration decreased to 28 days.

3) The remanufacturing cost added to total cost. It is calculated below:

100$ + (4)*10= $140

So now we need to use crashing to reduce the project duration for another 13 days to reach 15 days. The detail of it illustrated on the Table 4. 9.

Table 4.10 Information of each iteration in Case 3

Iteration	Activity	Normal Duration	Final Duration	Crash Cost
1,2,3	3	6	3	60
4,5	6	9	7	91
6	4	6	5	57.5
7,8	2	7	5	154
9	5	8	7	35.5
10	4	5	4	57.5

With a cursory glance at the table, we can see the activities are crashed 10 times. However, the total duration is decreased 9 time unit. It happens in iteration 9. Until this iteration, all the activities are crashed to the maximum possible time unit except activities 4 and 5. In these activities, by crashing one of them, the total duration does not change. However, if both activities are crashed by one unit, the makespan is decreased by one unit as well. So the algorithm chooses each of them to crash in iteration 9 and 10 separately. At the end, the algorithm could not crash the network more than 9 unit. So the total duration is 19 days. And we need to pay the penalty for the rest 4 days. The total crashing cost is $455.5, total penalty cost is $616, and the total cost is $1212.

Here the question is if crashing in iteration nine does not influence in total duration, so why algorithm do that? The answer to this question is because if we stop in iteration 8, it means we need to pay the penalty for 5 days instead of 4 days. It means another $154. But by crashing activity

5 and 4, we need to pay $93. As the objective function of this algorithm is the cost minimization, the algorithm choose the second option.

Case 4)

In this case, activities 4 and 5 are both remanufactured. So the changes are implemented as below:

1) Precedence relationship is changed. Here, the algorithm omitted activity 3. Because after remanufacturing activities 4 and 5, it is not a predecessors for any other activity. The new network is :

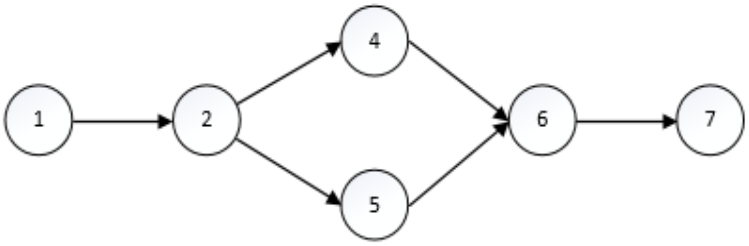

Figure 4.4 Network diagram in Case 4

2) In this case, both renewable resources which allocated to activity 3, will be ready again to use for other activities. Also, the usage of renewable type 2 (machines) for activities 4 and 5 are changed to zero. So it will be free to use for other activities. As a result, the total availability of renewable1 is changed from 10 to 12, and the total availability of renewable2 is changed from 7 to 20. The resource allocation for new network is:

Table 4.11 Resource allocation in Case 4

Day / Activity	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
2	*	*	*	*	*	*	*																	
4								*	*	*	*	*	*											
5								*	*	*	*	*	*	*	*									
6																*	*	*	*	*	*	*	*	*

As shown in Table 4.11, after remanufacturing activities 4 and 5, the total project duration decreased to 24 days.

3) The remanufacturing cost is added to the total cost. It is calculated below:

For activity 4: $100 + (3)*10= $130

For activity 5: $100 + (4)*10= $140

Total remanufacturing cost for both activities is equal to $130 + $140 = $270

Like explain in previous cases, we need to decrease the duration from 24 to 15 days if feasible. The detail of the each crashing iteration is illustrated in the Table 4.11.

Table 4.12 Information of each iteration in Case 4

Iteration	**Activity**	**Normal Duration**	**Final Duration**	**Crash Cost**
1,2	5	8	6	71
3,4	6	9	7	91
5,6	2	7	5	154
7	5	6	5	35.5
8	2	6	5	57.5
9	5	5	4	35.5
10	4	5	4	57.5

As shown above, the algorithm crashed activities in 10 iterations. However, the total duration is decreased 8 time unit. It happens in iteration 7 and 9. until these iterations, the total allowable crash time is used for all the activities except 4 and 5. To see changes in total duration we need to decrease the duration of activities 4 and 5 together because both of them are the predecessors for activity 6. However, as the logic of the algorithm is to reduce the duration of one activity in each iteration, so at first, it chooses activity 5 to decrease and then chooses activity 4. In total, the algorithm could not crash the network more than 8 time unit. So the total duration is 16 days. Hence we need to pay the penalty for the rest 1 day. The total crashing cost is $502, the total penalty cost is $154, and the total cost is $926. Until now, all four different cases described in detail. The graph below makes the compression between the total costs of all 4 cases.

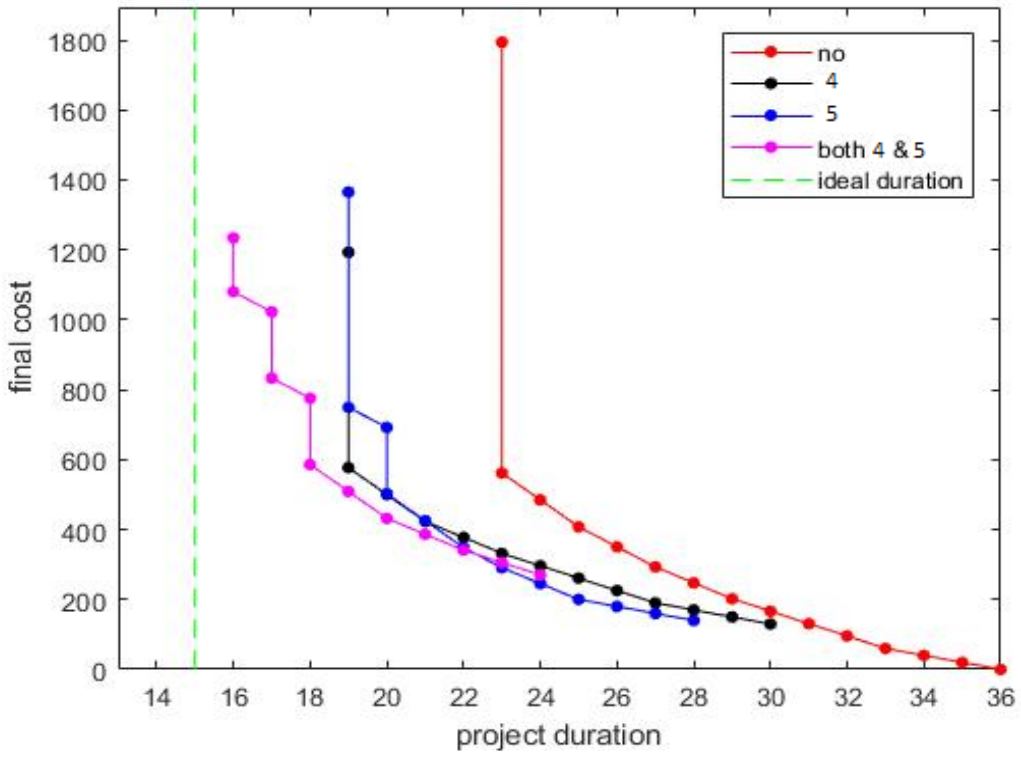

Figure 4.5 Total cost comparison between all Cases (15 days)

As the graph shown, remanufacturing has a significant influence on the total duration of the project. In this example remanufacturing alone, without crashing is decreased the total project duration by 25%. Also, it increases the chance of reaching the desired duration tremendously (case4). It also helps the flow of the network by omitting the bottleneck (changing in precedence relationship) in the project and makes more machines (renewable 2) free to use. Furthermore, it causes to pay less penalty cost (in case 4 we just pay the penalty for one day), so the total cost is decreased significantly.

4.3 Experimental Examples

For checking the efficiency of the algorithm, two different networks with different precedence relationship and resource usage are generated. The developed algorithm is tested on the instances, and the results are compared in each case. All the models are coded and solved by MATLAB.

The main goal of the algorithm is to minimize remanufacturing cost, crashing cost and penalty cost.

4.3.1 Example 1

In this example, the chosen network has 8 activities which 2 of them are dummy. Here the algorithm is tested under the circumstance that 2 successive activities are used remanufacturing option. The network and the information are shown in Figure 4.6 and Table 4.12 respectively.

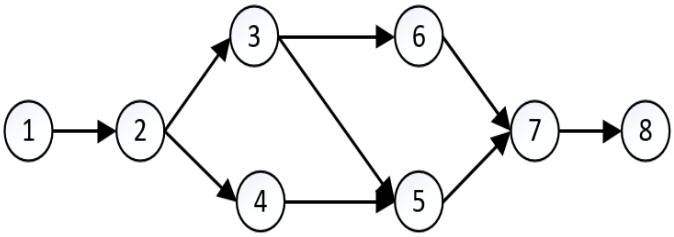

Figure 4.6 Network diagram in Case 1- example 1

Table 4.13 Network information for Example 1

Activity	Normal duration	Crash duration	Renewable 1(worker)	Renewable 2(machine)	Non-renewable	Normal cost	Crash cost
1	0	0	0	0	0	0	0
2	7	5	5	4	5	91	286
3	8	4	2	6	3	96	268
4	6	4	3	3	2	82	336
5	9	7	4	7	2	65	312
6	10	8	3	7	1	83	305
7	7	5	5	4	4	69	317
8	0	0	0	0	0	0	0

After running the algorithm in MATLAB, the summery of the result is:

Table 4.14 Result summery of example 1 (20 days)

	Initial Project Duration	Total Crashing Cost	Remanufacturing Cost	Penalty Cost	Total Cost	Final Duration	Desired Duration
Case 1	47	1338	0	3302	4640	33	20
Case 2	41	1338	130	2286	3754	29	20
Case 3	37	1338	110	1270	2718	25	20
Case 4	37	1338	240	1270	2848	25	20

In this case, remanufacturing reduce the makespan of the project by 18.5% on average without crashing. As the crashing costs are the same in all four different cases, it showed that remanufacturing reduce the total cost by 19% in case 2 and almost 42% in case 3 and 4. Also, it is realized that, when two successive activities have the option to use remanufacturing, it is better to use case 3. It means just remanufactured the successor activity.

The figure below make the comparison between them better:

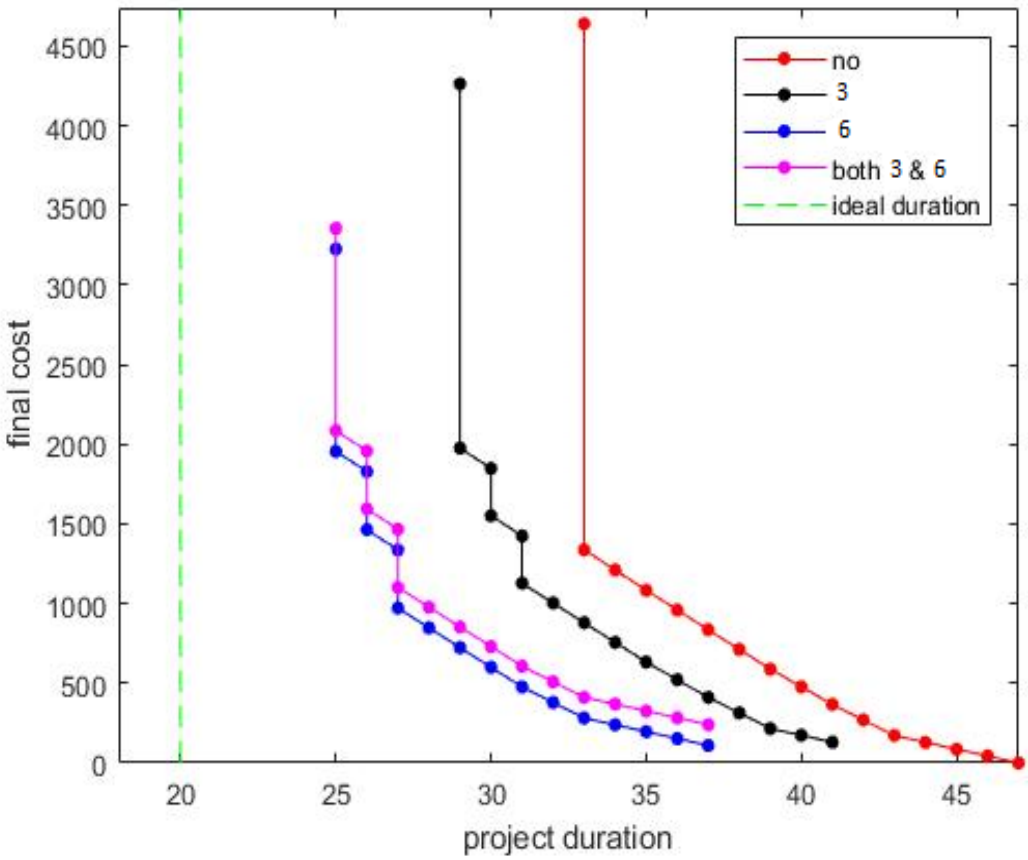

Figure 4.7 Total cost comparison between all cases in example1 (20 days)

Now we want to check the algorithm in the same network to see if the desired duration being nearer to the initial project duration, then what happened. The table below illustrate the summery of the result in this circumstance:

Table 4.15 Result summery of example 1 (25 days)

	Initial Project Duration	**Total Crashing Cost**	**Remanufacturing Cost**	**Penalty Cost**	**Total Cost**	**Final Duration**	**Desired Duration**
Case 1	47	1338	0	2032	3370	33	25
Case 2	41	1338	130	1016	2484	29	25
Case 3	37	1338	110	0	1448	25	25
Case 4	37	1338	240	0	1578	25	25

As table 4.14 shown, the crash cost is same as before. The only change happens in penalty cost which decreased in this condition significantly. But the important point is that remanufacturing still decreased the makespan of the project by 18.5% on average.

Also, it showed that remanufacturing still reduce the total cost by 19% in case2 and almost 42% in case 3 and 4.

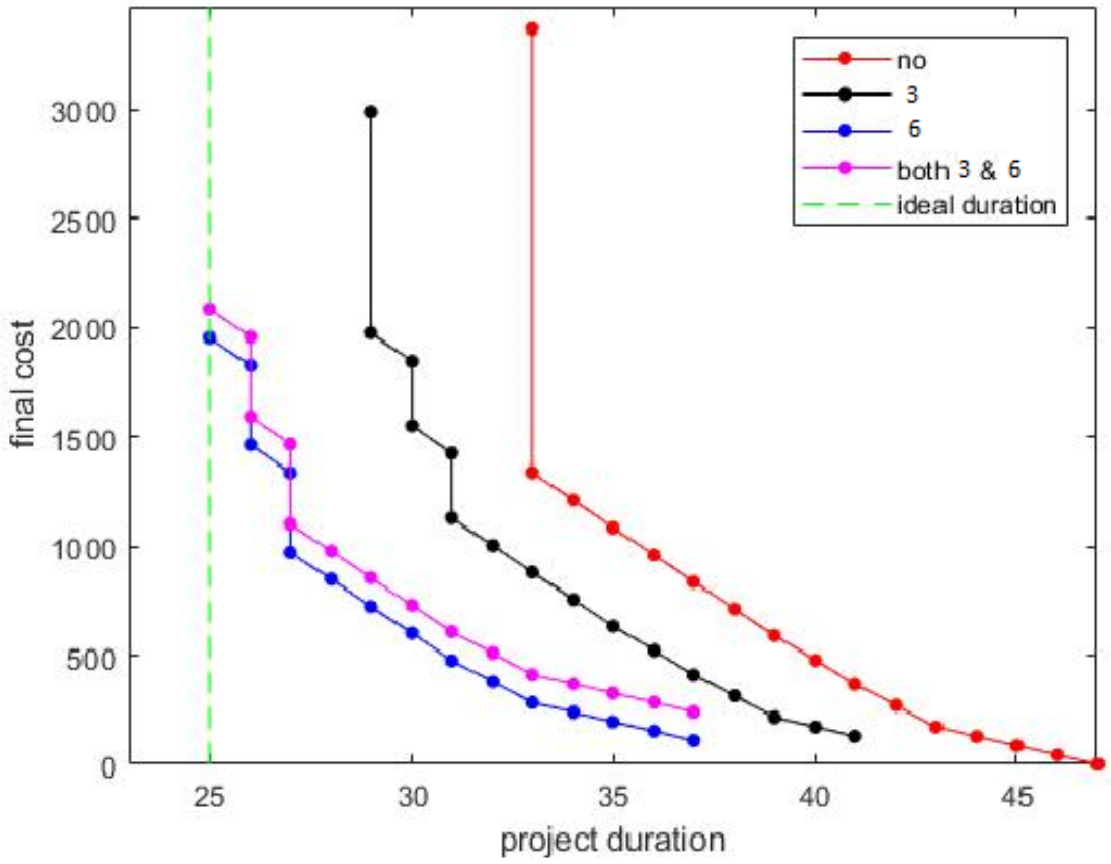

Figure 4.8 Total cost comparison between all cases in example1 (25 days)

Now, again decrease the discrepancy between initial duration and desired duration, we have:

Table 4.16 Result summery of example 1 (30 days)

	Initial Project Duration	**Total Crashing Cost**	**Remanufacturing Cost**	**Penalty Cost**	**Total Cost**	**Final Duration**	**Desired Duration**
Case 1	47	1338	0	762	2100	33	30
Case 2	41	1168	130	0	1298	30	30
Case 3	37	490.5	110	0	600.5	30	30
Case 4	37	490.5	240	0	730.5	30	30

Here, there is no need to pay any penalty cost in case 2, 3 and 4. Also, there is no need to crash all the activities to reach the desired duration. The algorithm in each case just choose the cheapest ones and crashed them until reached by day 30. As a result the crash cost also reduced a lot. However, the effect of remanufacturing still the same. It reduced the total project duration by 18.5% on average.

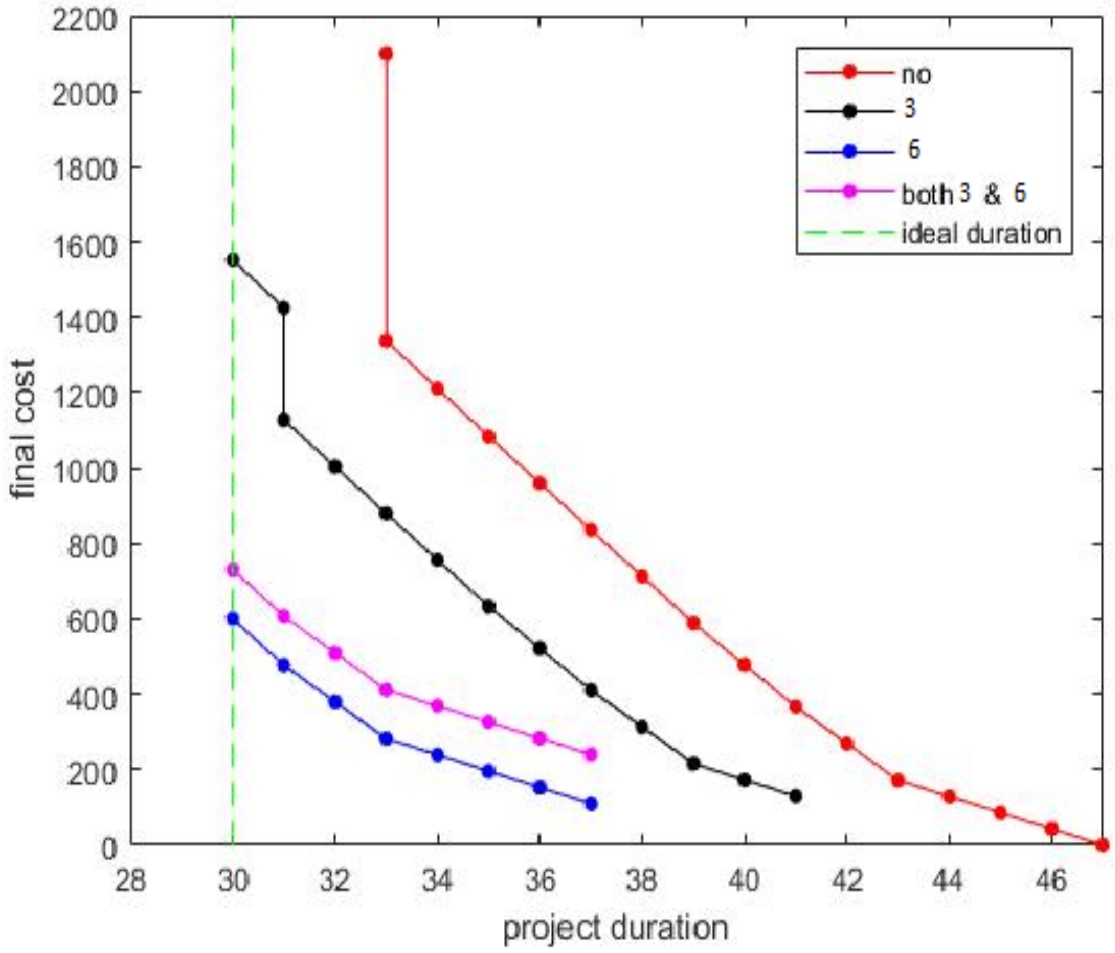

Figure 4.9 Total cost comparison between all cases in example1 (30 days)

The rest of analyses is done base on the information shown in figure 4.10. If we put the desired duration to 35 days, it still makes more sense to remanufactured activity 6 only and then crashed

the network for two more days with a total cost of $196. Let's assume we need to finish the project in 40 days. So it means just seven days less than the initial project duration. If we want to reach 40 days by paying crashing cost, we need to pay $490.5. However, by remanufacturing activity 6 we just pay $110, and instead of 40 days, the network crashed to 37 days. It means we finished the project three days sooner than the expected day. So we resaved bones for this three days. It is equal to 3 *154 = 462. It shows that we made $352 profit.

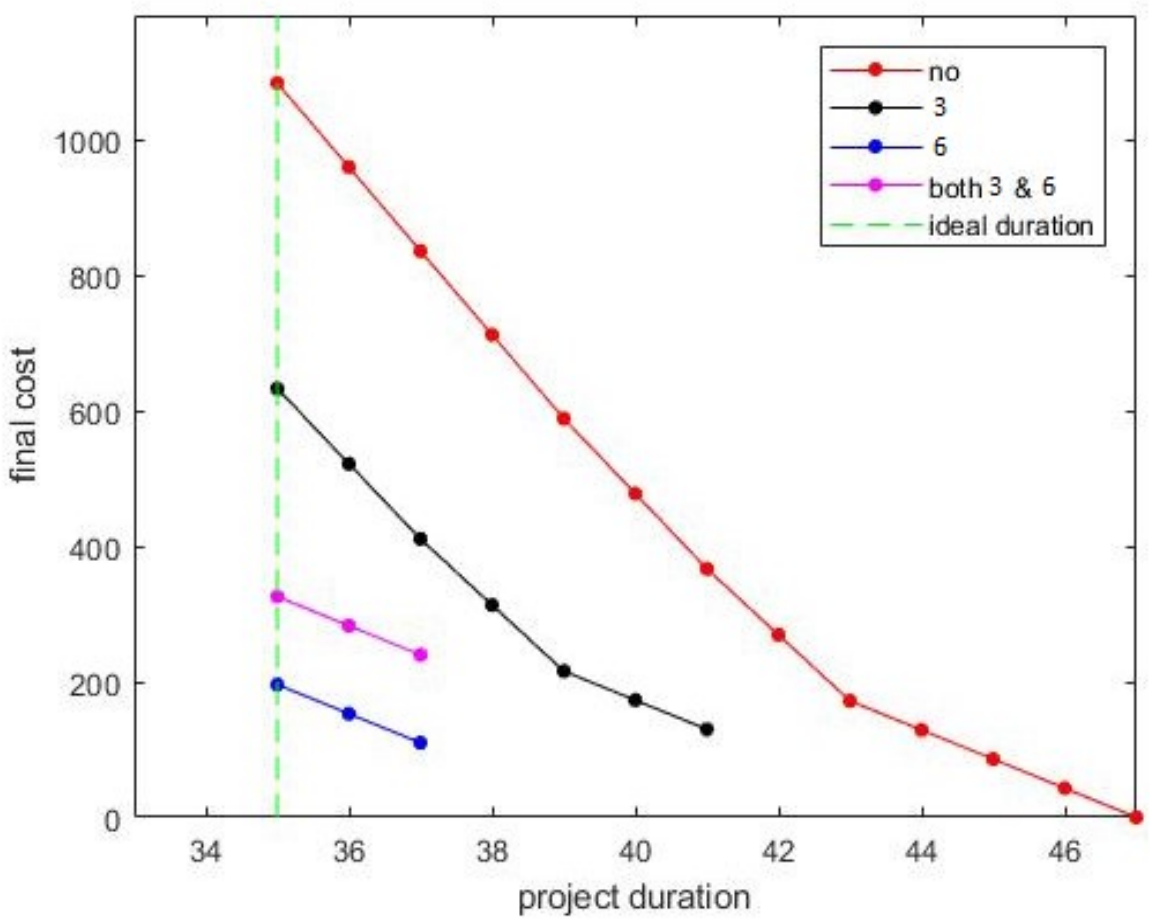

Figure 4.10 Total cost comparison between all cases in example1 (35 days)

4.3.2 Example2

Another example is generated to check the algorithm in another different network. Here, we have 9 activities which 2 of them are dummy. In this network it assumed activities 4 and 5 are remanufactured. The network and the rest of information regarding that are shown below:

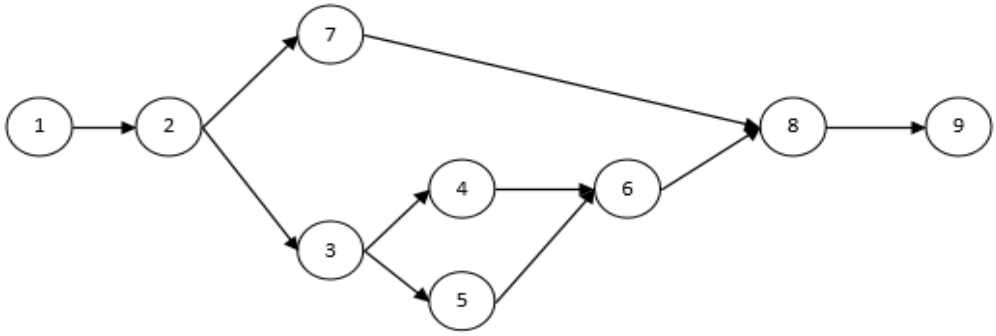

Figure 4.11 Network diagram for example 2 in Case 1

Table 4.17 Network information for example 2

Activity	Normal duration	Crash duration	Renewable 1(worker)	Renewable 2(machine)	Non-renewable	Normal cost	Crash cost
1	0	0	0	0	0	0	0
2	10	7	1	5	3	52	306
3	9	6	3	6	4	71	311
4	7	5	4	7	3	97	263
5	5	4	2	7	4	74	271
6	8	5	2	7	2	67	260
7	7	4	1	5	5	53	295
8	4	3	3	3	4	76	280
8	0	0	0	0	0	0	0

The project is solved in MATLAB and Table 4.17 show the summary of the results for each case.

Table 4.18 Result summery of example 2 (20 days)

	Initial Project Duration	**Total Crashing Cost**	**Remanufacturing Cost**	**Penalty Cost**	**Total Cost**	**Final Duration**	**Desired Duration**
Case 1	50	1496	0	5712	7208	34	20
Case 2	43	1496	130	3672	5298	29	20
Case 3	45	1496	140	4080	5716	30	20
Case 4	34	1256	270	1224	2750	23	20

Here, remanufacturing reduce the makespan of the project by 12% in average without crashing in case2 and 3, and 32% in case 4. Because in case 4, new network eliminate activity 3. It acted like omitted the bottleneck from the system. Consequently, the total project duration dropped significantly. The crashing costs are not the same in all 4 cases, because activity 3 deleted from the network. So it showed that remanufacturing reduce the total cost by 26.5% in case2 and 3, and 63% in case 4.

The Figure 4.12 show the comparison between costs of all cases.

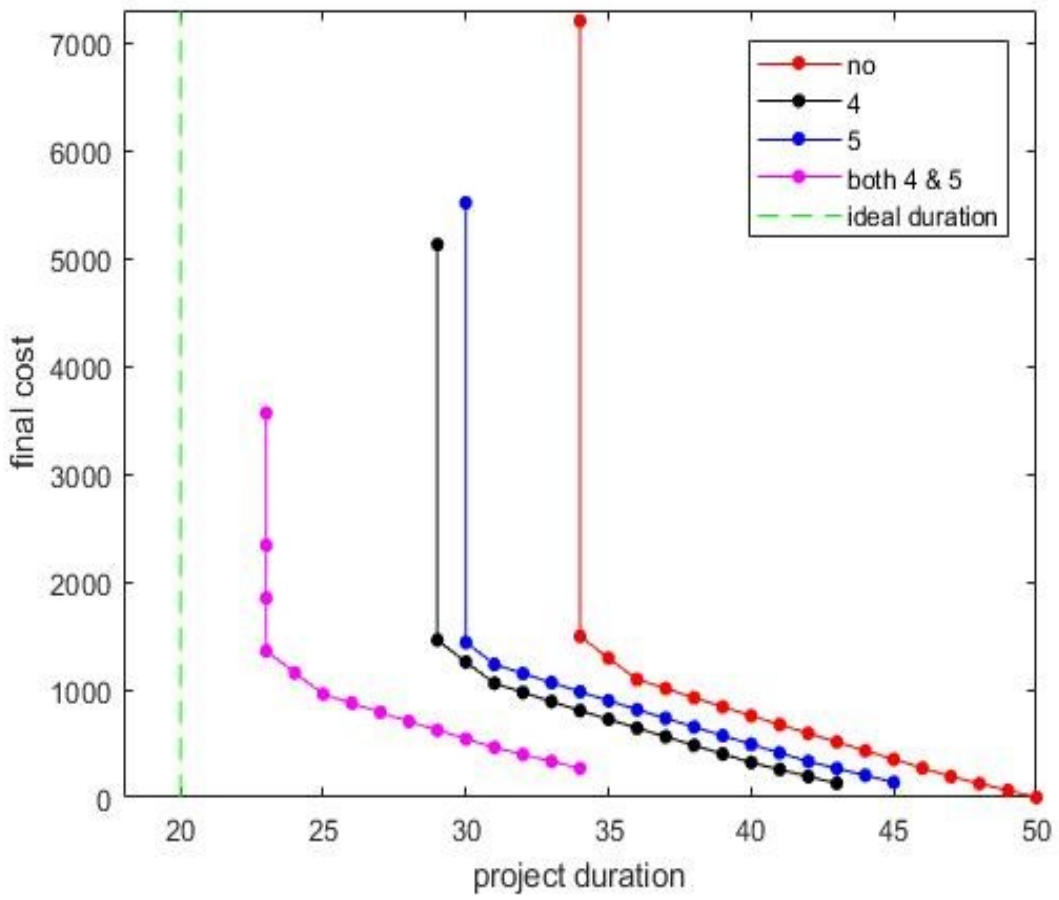

Figure 4.12 Total cost comparison between all cases in example2 (20 days)

Now we put the desired duration equal to 30 days. All the changes are illustrated in Table 4. 18.

Table 4.19 Result summery of example 2 (30 days)

	Initial Project Duration	**Total Crashing Cost**	**Remanufacturing Cost**	**Penalty Cost**	**Total Cost**	**Final Duration**	**Desired Duration**
Case 1	50	1496	0	1632	3128	34	30
Case 2	43	1126	130	0	1256	30	30
Case 3	45	1299	140	0	1439	30	30
Case 4	34	274	270	0	543.66	30	30

It still illustrate that remanufacturing both activities is the best choice for us.

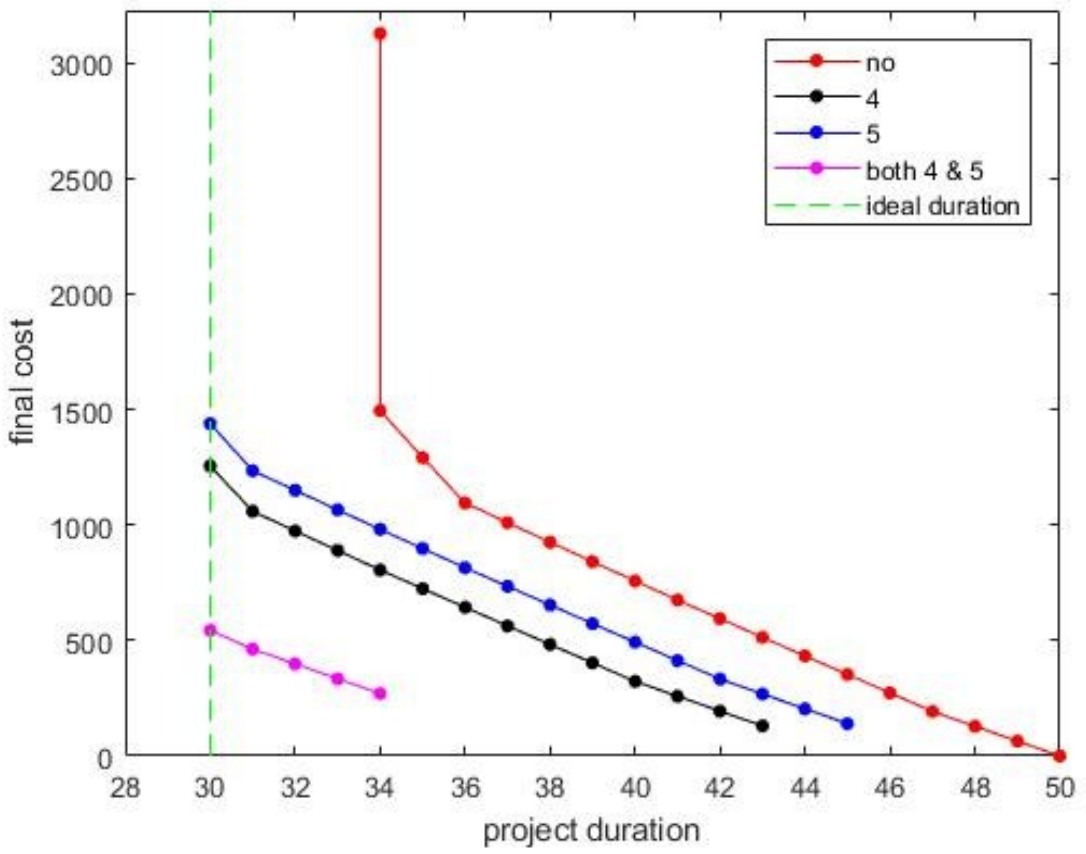

Figure 4.13 Total cost comparison between all cases in example2 (30 days)

The rest of information can realized from the figure 4.13. If we decreased the desired duration for another 10 days then we have still the option to remanufacture both activities 4 and 5. In this condition the total project duration reached to 34 days. It means we finished the project 6 days sooner than the desired duration. We just pay $270, however, we received the bones about $2448. As a result, still remanufacturing both activities is the best option.

4.4 Case Study

The presented algorithm is applied in a case study. The network used here is a part of the real project from a manufacturing company in northern Ontario. This is a project based company which produces and refurbishes train cars and locomotives. The chosen project has 278 activities in 9 stations. The selection network is related to the one of the station (station 5) of this project. In this station 5 different renewable resources are used. But we considered only two of them. The network is presented in Figure 4.14.

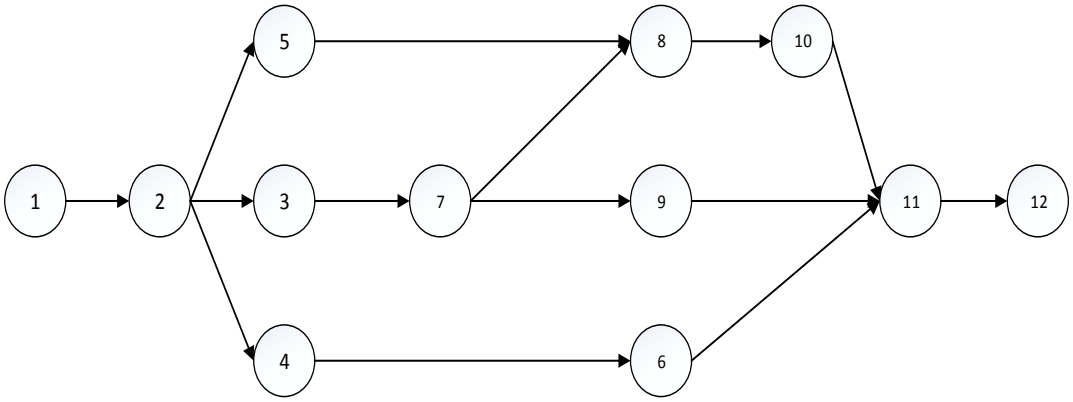

Figure 4.14 Network diagram for case study

In this station, activities 7 and 8 are capable of using remanufacturing. After implementing the algorithm, the result shows the different costs that company faced in four different cases. The presented network need 43 days to complete with the available resources. But to reach to final project deadline, the station 5 should be finished by day 30. The summary of the result is shown in Table 4.20.

Table 4.20 Result summery of case study

	Initial Project Duration	Total Crashing Cost	Remanufacturing Cost	Penalty Cost	Total Cost	Final Duration	Desired Duration
Case 1	43	8140	0	0	8140	30	30
Case 2	38	5842	140	0	5982	30	30
Case 3	38	6335	140	0	6475	30	30
Case 4	32	997	280	0	1277	30	30

As the table illustrated, in none of the cases penalty is paid. However, in case1 the discrepancy between the initial duration to desired duration is more than other cases. It will be costly for the company to cover this gap only by paying the crash cost for the activities. Hence, by activating the remanufacturing line, the total cost dropped significantly (especially in case 4).

In average, remanufacturing reduces the makespan of the project by 11.5% without crashing in cases 2 and 3, and 25.5% in case 4.

In case 4, new network eliminate activities 3 and 5. Because they were not the predecessor anymore. Consequently, the total project duration dropped significantly. The crashing costs are not the same in all 4 cases. However it showed that remanufacturing reduces the total cost almost by 27% in cases 2 and 3, and 84% in case 4.

Figure 4.15 makes the comparison between different cases:

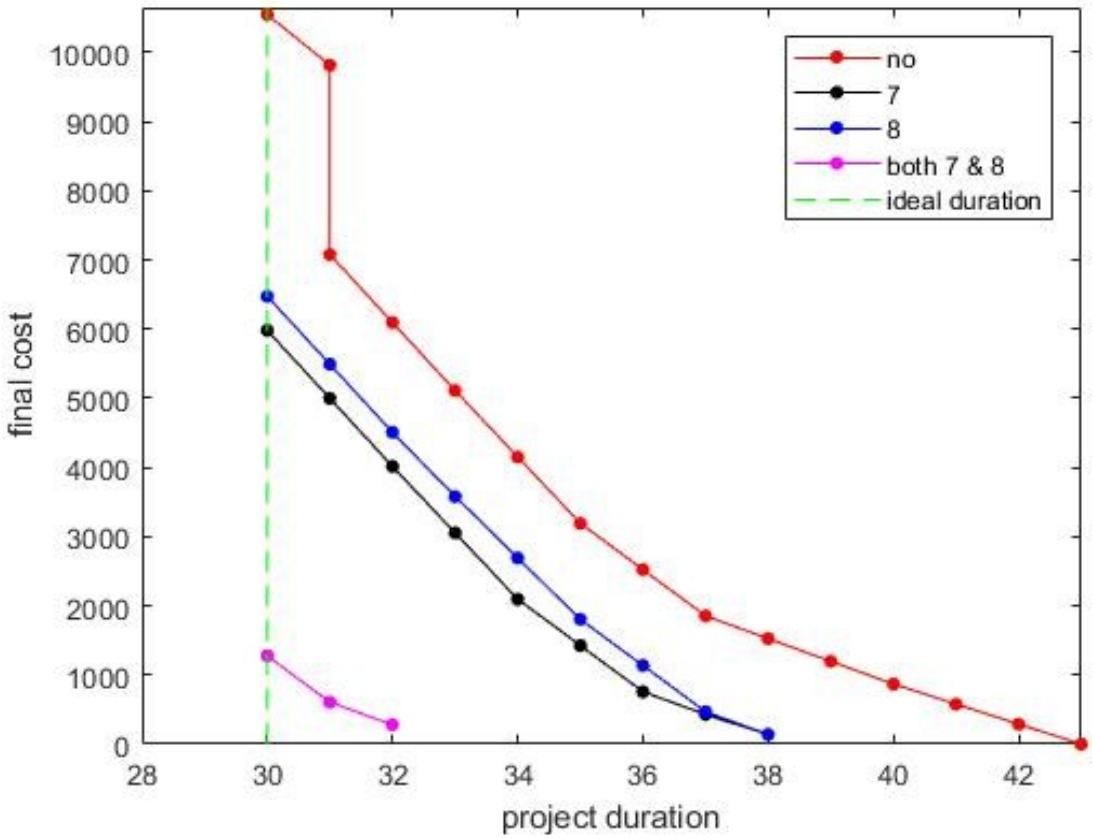

Figure 4.15 Total cost comparison between all cases in case study

4.5 Summary

In this chapter, at first, one example is illustrated in details. All the steps of the algorithm are described using numerical example. Then, two different instances with different networks and conditions are generated and solved with the algorithm. In this examples, as the results show, in all types of precedence relationship, remanufacturing has a significant impact on total cost of the project. Also, it helps the makespan to reach the desired duration by paying less amount of crash cost. At the end the algorithm implemented in the real industry situation as a case study. The result for this part show that by remanufacturing both possible activity, the total cost decrease 84%.

CHAPTER 5: CONCLUSION AND FUTURE WORK:

In this study, a new extension of project crashing problem was investigated. In this extension, Crashing Problem with Resource Constrained Scheduling and Remanufacturing (CPRCSR) was introduced in Chapter 3. A new mathematical model was proposed with considering Resource dedication policy. In Chapter 4 different examples were generated and solved with algorithm to test it in various circumstances. Also, the algorithm implemented in a case study as well.

Critical Path Method (CPM) for project crashing proposed in the literature cannot be applied when the resource limitation is introduced. The new algorithm deals with this issue. This algorithm uses two parts to reach an optimal solution. The first part uses a linear model to calculate the shortest makespan when all the activities can be completed with the limited resources. The model uses activate/deactivate concept for each activity in each time unit of the project to simplify resource allocation. In the second part, the algorithm uses the solution from the first part and tries to crash the project to reach the desired duration. In this part, some activities have option to use remanufacturing line. The algorithm tries to find the best combination between paying crash cost, activating remanufacturing and paying the penalty to reach the desired duration by minimizing the total cost of the project.

To test the proposed algorithm and compare it in different circumstances, three different examples and one case study were solved, and the results are compared. The algorithm is fully implemented in MATLAB.

In the future work, the computational complexity of the proposed approach can be optimized to deal with complex projects with many activities and dependencies. Additionally, the material rate

from remanufacturing considered deterministic, however, this assumption is not practical in many scenarios. A new method to take this into consideration can be studied in the future.

Appendix i

Main model Coded in MATLAB:

```
normalDuration          = […….];
crashDuration           = [……];
normalCost              = [……];
crashCost               = […….];
crashCostPerTimeUnit    = (crashCost - normalCost)./(normalDuration - crashDuration);
renewableResource1      = [……..];
renewableResource2      = [……];
maxRenewableResource1   = ….. ;
maxRenewableResource2   = …. ;
pSet                    = […….];  %PRESEDENCE RELATIONSHIP BETWEEN ACTIVITIES

numAct                  = length(normalDuration);
paths                   = pathsFinder(pSet,numAct,100);


% REMANUFACTURING OPTIONS
nonrenewableResource            = [……..];
possibleRemanufacturingOptions  = [……];
setupCost4Remanufacturing       = […..];
modificationCost4Remanufacturing = 10*nonrenewableResource(possibleRemanufacturingOptions);
remanufacturingCost             = setupCost4Remanufacturing + modificationCost4Remanufacturing;


% define the desired duration
desiredTotalDuration = … ;



% pack resource info in a structure
resource.renewableResource1  = renewableResource1;
resource.renewableResource2  = renewableResource2;
resource.maxAvailability     = [maxRenewableResource1, maxRenewableResource2];

% without remanufacturing
p{1} = pSet;
[solution{1}, totalCost(1), durations{1}, projectDurationAfterCrashing{1}, crashActHistory{1},
crashCostHistory{1}, durationHistory{1}] = crashingWithResrouceLimit(normalDuration, crashDuration,
crashCostPerTimeUnit, desiredTotalDuration, resource, pSet);




% with remanufacturing
ind = 2;
for activityNum = possibleRemanufacturingOptions


   resource.renewableResource2  = renewableResource2;
   resource.renewableResource2(activityNum) = 0;
```

```
    pSetNew = findNewPSetAfterRemanufacturing(paths, activityNum);
    [solution{ind}, totalCost(ind), durations{ind}, projectDurationAfterCrashing{ind}, crashActHistory{ind},
crashCostHistory{ind}, durationHistory{ind}] = crashingWithResrouceLimit(normalDuration, crashDuration,
crashCostPerTimeUnit, desiredTotalDuration, resource, pSetNew);
    totalCost(ind) = totalCost(ind) + remanufacturingCost(ind - 1);

    p{ind} = pSetNew;
    ind = ind + 1;
end


% both remanufactured
resource.renewableResource2   =  renewableResource2;
resource.renewableResource2(possibleRemanufacturingOptions) = 0;

pSetNew = findNewPSetAfterRemanufacturing(paths, possibleRemanufacturingOptions);
[solution{ind}, totalCost(ind), durations{ind}, projectDurationAfterCrashing{ind}, crashActHistory{ind},
crashCostHistory{ind}, durationHistory{ind}] = crashingWithResrouceLimit(normalDuration, crashDuration,
crashCostPerTimeUnit, desiredTotalDuration, resource, pSetNew);
totalCost(ind) = totalCost(ind) + sum(remanufacturingCost);
p{ind} = pSetNew;



meaningfulIndex = houseCleaning(projectDurationAfterCrashing, durationHistory);
for i = 1 : ind - 1
    solution{i} = solution{i}(1:meaningfulIndex(i));
    projectDurationAfterCrashing{i} = projectDurationAfterCrashing{i}(1:meaningfulIndex(i), :);
    crashActHistory{i} = crashActHistory{i}(1:meaningfulIndex(i),:);
    crashCostHistory{i} = crashCostHistory{i}(1:meaningfulIndex(i),:);
    durationHistory{i} = durationHistory{i}(1:meaningfulIndex(i),:);
end


delayPenaltyPerDay = max(crashCostPerTimeUnit)*2;
totalDelayPenalty = (cellfun(@(x) x.finalTotalDuration, durations) - desiredTotalDuration)*delayPenaltyPerDay;
totalCost = totalCost + totalDelayPenalty;




% plotting
remCost = [0 remanufacturingCost(1) remanufacturingCost(2) sum(remanufacturingCost)];
delCost = totalDelayPenalty;
startingPoint = durations{1}.initialTotalDurarion;
for selected = 1 : ind
    crashCost = crashCostHistory{selected};
    crashAct = crashActHistory{selected};
    clear temp
    temp(1) = remCost(selected);
    for i = 1 : size(crashCost,1)
        temp(i+1) = crashCost(i,crashAct(i));
```

```
    end

    info{selected} = cumsum(temp);
end


style = {'r.-','k.-','b.-','m.-'};
for i = 1 : 4
    h(i) = plot([durations{i}.initialTotalDurarion durationHistory{i}.'], info{i}, style{i}, 'markersize', 16);
    hold on
    plot([durationHistory{i}(end), durationHistory{i}(end)], [info{i}(end), info{i}(end) + delCost(i)], style{i},
'markersize', 16)
end
xlim([desiredTotalDuration - 2, durations{1}.initialTotalDurarion])
xlabel('project duration')
ylabel('final cost')
ymax = max(cellfun(@max, info)+delCost)+100;
h(5) = line([desiredTotalDuration desiredTotalDuration],[0 ymax],'Color','g','LineStyle','--');
ylim([0 ymax])
legend(h,'no', '4', '5', 'both 4 & 5', 'ideal duration')
```

Appendix ii

Called Function in main Code:

1) Path Finder:

```
function  paths = pathsFinder(pSet,nAct,maxIteration)


startIndexes = find( pSet(:,1) == 1);
for i = 1:length(startIndexes)
   pathsDictionary{i} = pSet(startIndexes(i),:);
end

for i = 1:maxIteration
   pathsTemp = findNextBranch(pathsDictionary{i},pSet,nAct);
   L = size(pathsDictionary,2);
   for k = 1:size(pathsTemp,2)
      pathsDictionary{L+k} = pathsTemp{k};
   end
   pathsDictionary{i} = [];


   if isSearchOver(pathsDictionary, nAct)
      paths = cleanPathsDictionary(pathsDictionary);
      break;
   end
end
```

2) Find New P Set after Remanufacturing:

```
function pSetNew = findNewPSetAfterRemanufacturing(paths, activities)

for activityNum = activities
   flags = cellfun(@(x) ~isempty(find(x == activityNum, 1)), paths);
   pathsWithRemanufacturingOption = find(flags);
   for i = pathsWithRemanufacturingOption
      temp = paths{i};
      ind = find(temp == activityNum);
      paths{i} = [temp(1) temp(ind:end)];
   end
end
```

```
nPaths = length(paths);
index  = 1;
for pathNum = 1 : nPaths
   temp = paths{pathNum};
   L = length(temp);
   for i = 1 : L - 1
      A(index,:) = [temp(i) temp(i+1)];
      index = index + 1;
   end
end
pSetNew = unique(A, 'rows', 'first');
```

3) House Cleaning:

```
function meaningfulIndex = houseCleaning(projectDurationAfterCrashing, durationHistory)


N = length(projectDurationAfterCrashing);
meaningfulIndex = nan(1,N);

for i = 1 : N
   thisProjectDurationCrashing = projectDurationAfterCrashing{i};
   thisProjectDurationCrashing(isnan(thisProjectDurationCrashing)) = 0;
   allNanRemoved = thisProjectDurationCrashing((sum(thisProjectDurationCrashing, 2) ~= 0),:);

   meaningfulCrashIteration = size(allNanRemoved,1);
   while( durationHistory{i}(meaningfulCrashIteration,:) == durationHistory{i}(meaningfulCrashIteration-1,:) )
      meaningfulCrashIteration = meaningfulCrashIteration - 1;
   end
   meaningfulIndex(i) = meaningfulCrashIteration;
end
```

4) Crashing With Resource Limit:

```
function [solution, totalCost, durations, projectDurationAfterCrashing, crashActHistory, crashCostHistory,
durationHistory] = crashingWithResrouceLimit(normalDuration, crashDuration, crashCostPerTimeUnit,
desiredTotalDuration, resource, pSet)


activitiesInvolved  =  unique(reshape(pSet.',1,2*size(pSet,1)));
numAct              =  length(normalDuration);
pastDuration        =  normalDuration;
[solution{1}, ~, totalDurarion]  =  resourceConstrainedSolution(pastDuration, resource, pSet);
initialTotalDurarion   =  totalDurarion;

projectDurationAfterCrashing = nan(initialTotalDurarion - desiredTotalDuration, numAct);
index = 1;
```

```
isFinished = false;
while desiredTotalDuration < totalDurarion && ~isFinished

   durationBeforCrasshing = totalDurarion;

   cost   = nan(1,numAct);
   isFinished = true;
   for actNum = activitiesInvolved
      base = zeros(1,numAct);
      base(actNum) = 1;
      currentDuration = pastDuration - base;

      if sum(currentDuration >= crashDuration) == numAct
         [solution{index + 1}, ~, totalDurarion] = resourceConstrainedSolution(currentDuration, resource, pSet);
         cost(actNum) = crashCostPerTimeUnit(actNum);
         projectDurationAfterCrashing(index, actNum) = totalDurarion;
         isFinished = false;
      else
         projectDurationAfterCrashing(index, actNum) = nan;          % remove the acitivity from the competition
      end
   end

   penalty                 =  max(crashCostPerTimeUnit)*2;
   numOfReductions         =  durationBeforCrasshing - projectDurationAfterCrashing(index, :);
   cost(numOfReductions == 0)  =  penalty + cost(numOfReductions == 0);                    % penelize it largely
as the project duration stays the same
   cost(numOfReductions > 0)   =  cost(numOfReductions > 0)./numOfReductions(numOfReductions > 0);   %
reward if the reduction is more than one unit

   if ~isFinished
      [~, n] = min(cost);
      currentDuration    = pastDuration;
      currentDuration(n) = currentDuration(n) - 1;
      pastDuration       = currentDuration;
      totalDurarion = projectDurationAfterCrashing(index, n);
      crashActHistory(index,:) = n;
      crashCostHistory(index,:) = cost;
      durationHistory(index,:) = totalDurarion;
      index = index + 1;
   end
end

finalTotalDuration  =  totalDurarion;
finalDuratrion      =  pastDuration;
totalCost           =  sum((normalDuration - finalDuratrion).*crashCostPerTimeUnit);


durations.initialTotalDurarion = initialTotalDurarion;
durations.finalTotalDuration = finalTotalDuration;
```

5) Find Best Activity To Crash:

```
function [currentActDuration, feasibleFlag, selectAct] = findBestActivitiesToCrash(paths, nAct,
normalActDuration, normalActCost, crashActDuration, crashActCost, currentActDuration)

feasibleFlag = 1;
f = (crashActCost - normalActCost)./(normalActDuration - crashActDuration);
f(isinf(f)) = 0;


mainSet = 1:nAct;
L = length(paths);

index = 1;
for i = 1:L
   base1 = zeros(1,nAct);
   base2 = zeros(1,nAct);
   temp = paths{i};
   for k = 1:length(temp)
      if currentActDuration(temp(k)) == crashActDuration(temp(k))
         base1(temp(k)) = 1;
      else
         base2(temp(k)) = 1;
      end
   end
   Aeq(index,:)     =  base1;
   Aeq(index+1,:)   =  base2;
   beq(index,1)     =  0;
   beq(index+1,:)   =  1;
   index = index + 2;
end


[x,~,EXITFLAG] = intlinprog(f,1:nAct,[],[],Aeq,beq,zeros(1,nAct),ones(1,nAct));

selectAct = find(x);

if EXITFLAG ~= -2
   currentActDuration = currentActDuration - x.';
else
   feasibleFlag = 0;
end
```

References

[1] A. Aghaie and H. Mokhtari, “Ant colony optimization algorithm for stochastic project crashing problem in PERT networks using MC simulation,” Int. J. Adv. Manuf. Technol, vol.45, pp.1051-1067, 2009.

[2] M. Baki, A. Chauch and W. Abdul-kader, “A heuristic solution procedure for the dynamic lot sizing problem with remanufacturing and product recovery,” Computers & Operations Research, vol.43, pp. 225-236, 2013.

[3] F. Boctor, "Heuristics for scheduling projects with resource restrictions and several resource-duration modes," *The International Journal of Production Research,* vol. 31, pp. 2547-2558, 1993.

[4] M. Bruni, L. Pugliese, P. Beraldi and F. Guerriro, “An adjustable robust optimization model for the resource-constrained project scheduling problem with uncertain activity durations,” Omega, vol. 71, pp.66-84, 2016.

[5] E. Demeulemeester, "Minimizing resource availability costs in time-limited project networks," *Management Science,* vol. 41, pp. 1590-1598, 1995.

[6] E. Demeulemeester, M. Vanhoucke and W. Herroelen, "RanGen: A random network generator for activity-on-the-node networks," *Journal of Scheduling,* vol. 6, *(1),* pp. 17-38, 2003.

[7] D. Debels, B. Reyck, R. Leus and M. Vanhoucke, “A hybrid scatter search/electromagnetism meta-heuristic for project scheduling,” *Eur. J. Oper. Res.,* vol. 169, pp. 638-653, 2004.

[8] A. Drexl and J. Gruenewald, "Nonpreemptive multi-mode resource-constrained project scheduling," *IIE Transactions,* vol. 25, pp. 74-81, 1993.

[9] M. Feylizadeh, M. Modarres, “Optimal Crashing of Multi Period-Multi Product Production Planning Problems,” World Applied Sciences, vol. 4, pp.499-505, 2008.

[10] G. Ferrer and D. Whybark, “Material Planning for a Remanufacturing Faculty,” Production and Operations Management, vol.10, pp. 112-124, 2001.

[11] D. Guide, “Production planning and control for remanufacturing: industry practice and research needs,” Journal of Operations Management, vol. 18, pp. 467-483, 1999.

[12] T. Gocken, “Solution of fuzzy multi-objective project crashing problem,” Neural Comput & Applic, vol.23, pp.2167-2175, 2012.

[13] S. Hartman, “Project Scheduling with Multiple Modes: A Genetic Algorithm,” Annals of Operations Research, vol. 102, pp. 111–135, 2001.

[14] A. Haga and T. O’keefe, “Crashing PERT Networks: A Simulation Approach,” Academy of Business and Administrative Sciences Conference, 2001.

[15] G. Hatcher, W. Ljomah and J. Windmill, “Design for remanufacture: a literature review and future research needs,” Journal of Cleaner Production, vol. 19, pp.2004-2014, 2011.

[16] J. Jozefowska, M. Mika, R. Rozycki, G. Waligora and J. Weglarz “Simulated Annealing for Multi-Mode Resource-Constrained Project Scheduling,” Annals of Operations Research, vol. 102, pp. 137–155, 2001.

[17] C. Kellenbrink and S. Helber, “Scheduling resource-constrained projects with a flexible project structure” Eur. J. Oper. Res., vol.246, pp.379–391, 2015.

[18] G. Knotts, M. Dror and C. Hartman, “Agent- based project scheduling,” IIE Transactions, vol. 32, pp. 387-401, 2007.

[19] E. Laan, M. Salomon and R. Dekker, “Inventory Control in Hybrid Systems with Remanufacturing,” Management Science, vol.45, pp.733-747, 1999.

[20] D. Lee, J. Kang and P. Xirouchakis, “Disassembly planning and scheduling: review and further research,” SAGE Proc Instn Mech Engrs journals, vol. 215, pp. 695-709, 2001.

[21] C. Li, F. Liu, H. Cao and Q. Wang, “A stochastic dynamic programming based model for uncertain production planning of remanufacturing system,” International Journal of Production Research, vol.47, pp. 3657-3668, 2010.

[22] S. Liu, “Fuzzy Activity Times in Critical Path and Project Crashing Problems,” Cybernetics and Systems, vol.34, pp. 161-172, 2003.

[23] A. Mohanty, J. Mishra and B. Satpathy, “Activity modes selection for project crashing through deterministic simulation,” Journal of Industrial Engineering and Management, vol.4, pp.610-623, 2011.

[24] S. Morgan and R. Gagnon, “A systematic literature review of remanufacturing scheduling,” International Journal of Production Research, vol.51, pp. 4853-4879, 2013.

[25] A. Mingozzi, V. Maniezzo, S. Ricciardelli and L. Bianco, "An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation," *Management Science,* vol. 44, pp. 714-729, 1998.

[26] K. Nakashima, H. Arimitsu, T. Nose and S. Kuriyama, “Optimal control of a remanufacturing system,” International Journal of Production Research, vol. 42, pp.3619-3625, 2007.

[27] P. Pulat and J. Horn, “Time-Resource Tradeoff Problem,” IEEE Transactions on Engineering Management, vol.43, 1996.

[28] A. Roemer and R. Ahmadi, “Concurrent Crashing and Overlapping in Product Development,” Operations Research and the Management Sciences, vol.52, pp. 606-622, 2004.

[29] F. Salewski, A. Schirmer and A. Drexl, "Project scheduling under resource and mode identity constraints: Model, complexity, methods, and application," *Eur. J. Oper. Res.,* vol. 102, pp. 88-110, 1997.

[30] S. Shan, Z. Hu, Z. Liu, J. Shi, L. Wang and Z. Bi, “An adaptive genetic algorithm for demand-driven and resource-constrained project scheduling in aircraft assembly,” Information Technology and Management, vol.18, pp 41–53, 2015

[31] R. Sonmez, M. Abbasi and F. Uysal “Critical Sequence Crashing Heuristic for Resource-Constrained Discrete Time–Cost Trade-Off Problem,” Journal of Construction Engineering and Management, vol. 142, pp. 04015090, 2015

[32] R. Słowiński, "Two approaches to problems of resource allocation among project activities—a comparative study," *J. Oper. Res. Soc.,* vol. 31, pp. 711-723, 1980.

[33] S. Tao and Z. Dong, “Scheduling resource-constrained project problem with alternative activity chains,” Computers & Industrial Engineering 114, pp.288–296, 2017.

[34] M. Tritschler, A. Naber and R. Kolisch, (2017), “A hybrid metaheuristic for resource-constrained project scheduling with flexible resource profiles,” *Eur. J. Oper. Res.,* vol. 262, pp. 262-273, 2017.

[35] V. Van Peteghem and M. Vanhoucke, "An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances," *Eur. J. Oper. Res.,* vol. 235, pp. 62-72, 2014.

[36] T. Yang, “Performing complex project crashing analysis with aid of particle swarm optimization algorithm,” International Journal of Project Management, vol. 25, pp.637-646, 2006.

[37] S. Zanoni, I. Ferretti and O. Tang, “Cost performance and bullwhip effect in a hybrid manufacturing and remanufacturing system with different control policies,” International Journal of Production Research, vol. 44, pp.3847-3862, 2011.

[38] S. Zhou, Z. Tao and S. Tang, “Managing a Remanufacturing System with Random Yield: Properties, Observations, and Heuristics,” Production and Operations Management, vol.21, pp.797-813, 2012.