Spring 2017

Multi-Mode Resource Constrained Project Scheduling Using Differential Evolution Algorithm

Faisal Manour Altarazi
Old Dominion University, tarazifm@yahoo.com

MULTI-MODE RESOURCE CONSTRAINED PROJECT SCHEDULING USING DIFFERENTIAL EVOLUTION ALGORITHM

by

Faisal Manour Altarazi
B.S. August 2004, Riyadh College of Technology, Saudi Arabia
M.S. December 2010, Gannon University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for Degree of

DOCTOR OF PHILOSOPHY

DEPARTMENT OF MECHANICAL & AEROSPACE ENGINEERING

OLD DOMINION UNIVERSITY
May 2017

Approved by:

Han Bao (Director)

Rafael Landaeta (Member)

Xiaoyu Zhang (Member)

ABSTRACT

MULTI-MODE RESOURCE CONSTRAINED PROJECT SCHEDULING USING DIFFERENTIAL EVOLUTION ALGORITHM

Faisal Manour Altarazi
Old Dominion University, 2017
Director: Dr. Han Bao

Project scheduling is a tool that manages the work and resources associated with delivering a project on time. Project scheduling is important to organize, keep track of the finished and in-progress tasks and manage the quality of work delivered. However, many problems arise during project scheduling. Minimizing project duration is the primary objective. Project cost is also a critical matter, but there will always be a trade off between project time and cost (Ghoddousiet et al., 2013), so scheduling activities can be challenging due to precedence activities, resources, and execution modes. Schedule reduction is heavily dependent on the availability of resources (Zhuo et al., 2013).

There have been several methods used to solve the project scheduling problem. This dissertation will focus on finding the optimal solution with minimum makespan at lowest possible cost. Schedules should help manage the project and not give a general estimate of the project duration. It is important to have realistic time estimates and resources to give accurate schedules. Generally, project scheduling problems are challenging from a computational point of view (Brucker et al., 1999).

This dissertation applies the differential evolution algorithm (DEA) to multi mode, multi resource constrained project scheduling problems. DEA was applied to a common 14-task network through different scenarios, which includes Multi Mode Single Non Renewable Resource Constrained Project Scheduling Problem (MMSNR) and Multi Mode Multiple Non Renewable Resource Constrained Project Scheduling Problem (MMMNR). DEA was also applied when each scenario was faced with a weekly constraintand when cost and time

contingencies such as budget drops or change in expected project completion times interfere with the initial project scheduling plan. A benchmark problem was also presented to compare the DEA results with other optimization techniques such as a genetic algorithm (GA), a particle swarm optimization (PSO) and ant colony optimization (ACO). The results indicated that our DEA performs at least as good as these techniques as far as the project time is concerned and outperforms them in computational times and success rates. Finally, a pareto frontier was investigated, resulting in optimal solutions for a multi objective problem focusing on the tradeoff of the constrained set of parameters.

I dedicate this dissertation to my loving parents Khadija Ahmad Bukhari and Manour Mahmood Altarazi who guided me with their prayers. I also dedicate this dissertation to my supportive wife Arwa and my children Mariah, Sarah and Abdallah who were my motivation. Also, to my siblings Suad, Raed, Mohammad, Daniah and Sultan and everyone who supported me throughout this journey.

ACKNOWLEDGMENTS

I would like to express my gratitude to Dr. Han Bao for his support and help. He guided me through my courses and research, and gave advice and encouragement throughout my degree.

I would also like to thank Dr. Rafael Landaeta and Dr. Xiaoyu Zhang for being on my committee. I will always owe a great deal of gratitude toward these professors who provided guidance, knowledge and advice.

NOTATIONS

DEA	Differential Evolution Algorithm
MM	Multiple Mode
SNR	Single Non-Renewable Resource
MNR	Multiple Non-Renewable Resources
i	Vector's Population
j	Task
g	Generation
C_i	Feasible Solution Vector *i*
C_r	Crossover Factor
CP	Critical Path
C_{max}	Critical Path Total Project Time
s_i	Sequence Vector
m_i	Mode Vector
A	Scale Factor to Control Evolution Rate
rand	Random Number betwen 0 and 1
$r_{i,j}$	Random Generated Number for Sequence or Mode *i* and task *j*
$M_{c,s}$	Mutation Vector for Task C, Sequence Vector
$M_{c,m}$	Mutation Vector for Task C, Mode Vector
N_l	Units of Resources for Resource Type *l* Available
$Tg_{i,j}$	Target Vector for Sequence or Mode *i* and task *j*
$Tr_{i,j}$	Trial Vector for Sequence or Mode *i* and task *j*
TPT	Total Project Time
RCPSP	Resource Constrained Project Scheduling Problem
MRCPSP	Multi-mode Resource Constrained Project Scheduling Problem
MMRCPSP	Multi-mode Multi Resource Constrained Project Scheduling Problem
GA	Genetic Algorithm
SA	Simulated Annealing
PSO	Particle Swarm Optimization
ACO	Ant Colony Optimization

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

Projects are unique in nature. Time, cost and resources are normally considered when scheduling a traditional project. Project scheduling problems typically specify the minimization of project duration as the primary objective. However, cost is also a critical matter. Minimizing project time and cost is important in projects but it will have an influence on the project quality and risk (Zhou et al., 2013), and will always be a trade off between project time and cost (Ghoddousi et al., 2013). Scheduling activities can be challenging due to precedence activities, resources, and execution modes. Schedule reduction is heavily dependent on the availability of resources (Zhuo et al., 2013). This leads to the resource-constrained project scheduling problem (RCPSP) that was first introduced by Kelly in 1963. RCPSP concentrates scheduling activities over time and resources simultaneously based on the precedence that optimizes the scheduling objective, minimizing the project makespan.

Resources may be renewable or non-renewable. Renewable resources are used up in each period but reappear again at the beginning of the next period or when the task or tasks, which use those resources, are complete. Examples of renewable resources include manpower and many types of equipment and machines. Non-renewable resources are depleted as they are used and are available on a total project basis. Examples of non-renewable resources include capital, energy and raw materials. Any task may require a single resource or a set of resources, and the resource usage may vary over the duration of the task. A task may also have multiple execution modes (Sprecher, 1994).

The extension of RCPSP is the multi-mode resource constrained project scheduling problems (MRCPSP) which are more common in the real world where each activity is executed in one of several modes, “M1 (regular), M2 (fast), M3 (extreme)”, representing a combination of resources and durations. It is a challenging problem, and several techniques

have been proposed to solve this problem. MRCPSP are considered combinatorial problems, the optimum solution of which can be theoretically determined through finite steps (Mori & Tseng, 1997).

RCPSP and MRCPSP assume that once an activity starts, it will be executed until its completion (Peteghem & Vanhoucke, 2010). However, in resource-constrained project scheduling problems, the tasks have resource requirements and the resources are limited. In multi-mode resource-constrained project scheduling problems, each task may be executed in more than one mode, and each mode may have different resource requirements.

Problems need to be explicitly formulated (i.e. the objective function and constraints) (Zhuo et al., 2013). There have been several methods that were used to solve the project scheduling problem. This dissertation will focus on finding the optimal solution with minimum makespan at lowest possible cost. In addition, it is attempting to determine a contingency plan when a project faces a customer's request for a new delivery time and/or a change in budget constraints.

CHAPTER 2

LITERATURE REVIEW

2.1 Project Scheduling

Project scheduling problems are focused on finding the feasible optimal solution by investigating different execution modes with minimum makespan. The program evaluation and review technique (PERT) and critical path method (CPM) were widely used methods for project planning and scheduling (Lancaster & Ozbayrak, 2007). They were developed in the late 1950s (Kelly & Walker, 1959). Activities on the critical path are considered critical activities. CPM can determine the shortest possible time to complete the project (Zhou et al., 2013) by using the estimated task durations without considering probabilities (Lewis, 2011). It takes into account the time and determines critical activities to minimize project makespan, but resource availability is not considered, and an activity can start when all predecessor activities are completed. This is impractical because in a real project, resource availability and allocation will affect the entire project scheduling. The effect of resource constraint on project duration is very important (Chen & Zhou, 2013). To overcome CPM limitations, several techniques and optimizations have been proposed in project scheduling (Ghoddousi et al., 2013).

Usually a schedule is developed under the assumption of unlimited resources (Lewis, 2011). In reality real constraints such as those due to limited resources force the schedule to be modified before becoming a practical one. However, schedules should help to manage the project and not give a general estimate of the project duration. It is important to have realistic time estimates and resources to give accurate schedules. Generally, project scheduling problems are challenging from a computational point of view (Brucker et al., 1999).

2.2 Fundamentals of Project Scheduling

Over the last 4 decades, many books have appeared that focus on sequencing and scheduling; here is a brief review of books on sequencing and scheduling:

- **Muth & Thompson (1963):** This book contains a collection of papers focusing on computational aspects of scheduling.
- **Conway, Maxwell & Miller (1967):** This book deals with some of the stochastic aspects and priority queues.
- **Baker (1974):** This source gives an excellent overview of the many aspects of deterministic scheduling. This book does not deal with computational complexity issues since it appeared just before research in computational complexity started to become popular.
- **Coffman (1976):** This book is a compendium of papers on deterministic scheduling and covering computational complexity.
- **French (1982):** This covers most of the techniques that are used in deterministic scheduling techniques.
- **Dauzere-Peres & Lasserre (1994):** This source focuses primarily on job shop scheduling.
- **Brucker (1995):** This book presents a very detailed algorithmic analysis of many deterministic scheduling.
- **Pinedo & Chao (1999):** This source is more application oriented and describes a number of different scheduling models for problems arising in manufacturing as well as services.
- **Neumann, Schwindt & Zimmermann (2002):** This book covers basic concepts on project scheduling, resource constrained project scheduling with minimizing project duration and non-regular objectives.

- **Sule (2008):** This book provides a broad outlook on optimization and planning from the initial stages in the area of industrial scheduling and sequencing.
- **Lewis (2010):** This book presents real-world examples and provides applications-oriented understanding on project planning, scheduling and control.
- **Wilson (2014):** This source presents principles and techniques of scheduling and cost control. It focuses on the specific principles, techniques, and best-practice methodologies of scheduling and cost control.

2.3 Multi-Mode Multi Resource Constrained Project Scheduling Problems

Multi-mode multi resource constrained project scheduling problems (MMRCPSP) are more common in the real world. Each activity can be executed in one of a set of modes. Once the activity starts the selected mode cannot be changed. The objective is to find a minimal makespan schedule that meets the constraints imposed by the precedence relations and by the limited resources available (Brucker et al., 1999).

Several approaches have been proposed to solve the MMRCPSP such as the branch and bound proposed by Bruker et al. (1998) and Sprecher and Drexl (1998). However, the branch and bound is not able to solve large realistic projects since they cannot find the optimal solution in reasonable computation time (Peteghem & Vanhoucke, 2010).

Integer Programming (IP)/ Linear programming (LP) is a mathematical method for solving the optimization problem with linear objective functions subject to linear equality and inequality constraints. Mathematical methods for scheduling have received a considerable amount of attention due to their innate efficiency and accuracy. A disadvantage of this method is that the computational burden may grow tremendously as the problem size increases. In addition, this method has a single focus (leveling the resources); thus, the maximization of production rates is not considered (Zhou et al., 2013).

Heuristic methods are non-computer approaches that require less computational effort than mathematical methods. Traditional heuristic methods can only optimize one objective, and a global optimum is not guaranteed. The advantage of heuristic methods is their simplicity. However, most heuristic methods are problem dependent, which makes it difficult to apply them to other projects equivalently (Zhou et al., 2013).

Metaheuristic methods are used for solving combinatorial optimization problems by mimicking certain natural processes. Bouleiman and Lecocq (2003) and Józefowska et al. (2001) used the simulated annealing (SA) approach to solve MRCPSP. Jarboui et al. (2008) presented the particle swarm optimization (PSO). Genetic algorithm (GA) and ant colony optimization (ACO) are also methods to ensure optimal solutions (Zhou et al., 2013).

Genetic algorithms introduced by Holland (1975) use techniques and procedures inspired by the biological theory of evolution to solve complex optimization problems. Mori and Tseng (1997) proposed a genetic algorithm for MMRCPSP. Hartmann (2001) and Alcaraz et al. (2003) worked on a genetic algorithm for solving MMRCPSP. The genetic algorithm differs from the other meta-heuristic techniques (such as simulated annealing or tabu search) by producing a population of solutions rather than a unique current solution (Lancaster & Ozbayrak, 2007).

Evolutionary algorithms have been developed based on a form of meta-heuristic techniques especially by genetic algorithms. Evolutionary algorithms have shown to be well suited for complex problems. Project scheduling problems are distinctly complex and would benefit from evolutionary techniques for finding optimal solutions or near optimal solutions (Lancaster & Ozbayrak, 2007). One of the large algorithms developed in the domain of evolution was differential evolution (DE) introduced by Storn and Price in 1997. Because this dissertation relies extensively on the application of this method to solving project scheduling problems, it will be discussed at length later.

2.4 Common Software in Project Scheduling

Currently there are many commercially available software packages used for project scheduling. A listing of these software is provided below.

Microsoft Project: Is a project management software program developed and sold by Microsoft. It is very easy to learn, designed to assist a project manager in developing a plan, assigning resources to tasks, tracking progress, managing the budget and analyzing workloads.

Add ons to MS Project

- **Concerto:** Is critical chain project management software that requires a realization-consulting contract on top of the software costs (Concerto Integrated Software Solutions, 2016).
- **CC- (M) Pulse**: Uses the concept of critical chain methodology to create project plans. It is open source software and download is free (CC- (M) Pulse, 2016).
- **Pro-Chain:** Pro-Chain Project Management Solutions enhances the office software MS Project. There is an enterprise version, which acts as a database engine for the data (ProChain Solutions Inc, 2016).
- **PD-Trak:** PD-Trak Solution Software is an add-on for MS Project. User interface appears like someone wrote it in MS Access based on the website's screen shots (PD-Trak, 2016).

Primavera: Is a powerful and easy to use solution for planning and executing projects. It can plan, schedule and control complex projects, allocates best resources and tracks progress (Oracle's Primavera P6 Professional Project Management, 2016).

PS8 (used by Newport News Shipyard): Is a robust yet scalable project management tool. The program enables you to save time with its powerful resource leveling that you can use to keep your task plans viable given your resource constraints. Its leveling algorithm supports

single- and multi-project leveling using time-phased, resource-availability profiles (Project Management Software, 2016).

CC based software

- **Agile CC from AdeptTracker:** Is single project/ multi project CCPM software. The user can backward schedule a plan, identify the critical chain, size and place both the project and feeding buffer, size and place capacity buffer (AgileCC for AdeptTracker, 2008).

Corporate Systems

- **Siemen's PLM:** Product lifecycle management (PLM) is an information management system that can integrate data, processes, business systems and, ultimately, people in an extended enterprise. PLM software allows you to manage this information throughout the entire lifecycle of a product efficiently and cost-effectively, from ideation, design and manufacture, through service and disposal (Siemens PLM Software, 2016).
- **SAP:** The original SAP idea was to provide customers with the ability to interact with a common corporate database for a comprehensive range of applications. It has the capability to manage financial, asset, and cost accounting, production operations and materials, personnel, plants, and archived documents (Payne et al., 2014).

CHAPTER 3

DIFFERENTIAL EVOLUTION ALGORITHM WITH MULTI-MODE MULTI RESOURCE CONSTRAINED PROJECT SCHEDULING PROBLEM

3.1 Principles of DEA

Some problems are difficult if not impossible to solve. Differential Evolution (DE) can be used to find an approximate solution to such problems. It is a well-known scheme for global optimization and is a powerful heuristic method for solving multi-objective optimization problems. Based on the manipulation of random numbers, it is inspired by the genetic algorithm. DE has helped solve many industrial problems over the last ten years. DE is based on four main steps: population structure, mutation, crossover and selection.

Population Structure: The initial population of feasible solutions is randomly generated which conforms to their precedence constraints. Each feasible solution is represented by a vector of attributes associated with each task involved in the project. For example, one attribute may be the task's sequence. Another attribute may be the task's mode of operation.

Mutation: Two parents from the population are used to create a child, *M*, using some type of numeric function.

Crossover: The idea of the crossover is to create a trial vector from the target vector and the mutant vector by crossing them over.

Selection: It will be performed between the target vector and the trial vector, keeping the better individual and discarding the worse one.

The principles above will be illustrated in the case study given in section 3.2 below.

3.2 A Case Study of Multi-Mode Multi Resource Constrained Project Scheduling Problem

Damak et al. (2009) proposed a DE algorithm to solve MMRCPSP. The case study shown in Figure 3.1 involves six tasks that can be executed in either of the two modes as indicated in Table 3.2. Each sequence of a task is feasible only if it follows the precedence rule. The objective is to select the sequence with a makespan as small as possible. A solution is represented by two vectors: a position vector, which refers to the position of each task in the sequence, and a mode vector, which indicates the corresponding mode of each task. An example of a feasible solution is shown in Table 3.1 where the task sequence and related mode are shown for each of the 6 tasks in the network.

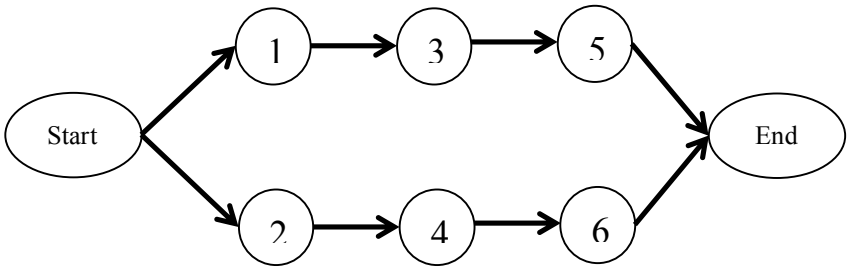


Figure 3.1: Damak et al. Example Network

Table 3.1: Representation of a Feasible Solution

Task	1	2	3	4	5	6
Task Sequence	1	4	2	5	3	6
Task Mode	2	1	2	2	1	1

Damak et al. explained the differential evolution steps for one generation manually and did not present the results. In this dissertation, a new case study will be introduced, and the differential evolution algorithm will be used to present the optimal solution through different scenarios using Matlab.

Table 3.2: Resource Requirements and Corresponding Durations for Two Modes

Task	Mode 1		Mode 2	
	Consumption NR/NN	Duration	Consumption NR/NN	Duration
1	2/4	3	1/2	4
2	3/4	4	2/3	6
3	4/2	2	2/2	3
4	4/6	2	3/3	3
5	3/1	1	1/5	3
6	2/1	4	1/1	6

Note. From "Differential evolution for solving multi-mode resource-constrained project scheduling problems" by N. Damak, B. Jarboui, P. Siarry and T. Loukil, 2009, Journal of Computers & Operation Research, 36, p. 2655.
NR= Renewable Resources
NN=Non-Renewable Resources

3.3 Research Gaps

The case study presented in Damak's paper is an important demonstration of the power of DE in providing an optimum solution for a project involving 6 tasks restricted by both renewable and non-renewable resources and by the number of modes available to each task. Nevertheless, a number of gaps can be identified as explained below.

- **Multi-Mode Project Scheduling**

  Damak et al. presented a case study with only two modes. This research extends the application of PSP to more than 2 modes.

- **Multi-Resource Project Scheduling**

  Damak et al. presented a case study with only two resources. This research extends the application of PSP to more than 2 resources.

- **Impact of Contingency Event**

  The most important part of this dissertation is investigating project scheduling with weekly constraints and changes in budget and comparing DEA with a benchmark problem. In addition, it develops a pareto frontier, which is an important and practical tool to test tradeoff between cost and time.

CHAPTER 4

APPLICATIONS OF DEA TO VARIOUS SCENARIOS

4.1 General Network for All Scenarios

The case study shown in Figure 4.1 involves fourteen tasks that can be executed in either of the three modes as indicated in Table 4.1. Each sequence of tasks is feasible only if it follows the precedence rule. Two vectors represent a solution: a position vector, which refers to the position of each task in the sequence, and a mode vector, which indicates the corresponding mode of each task.

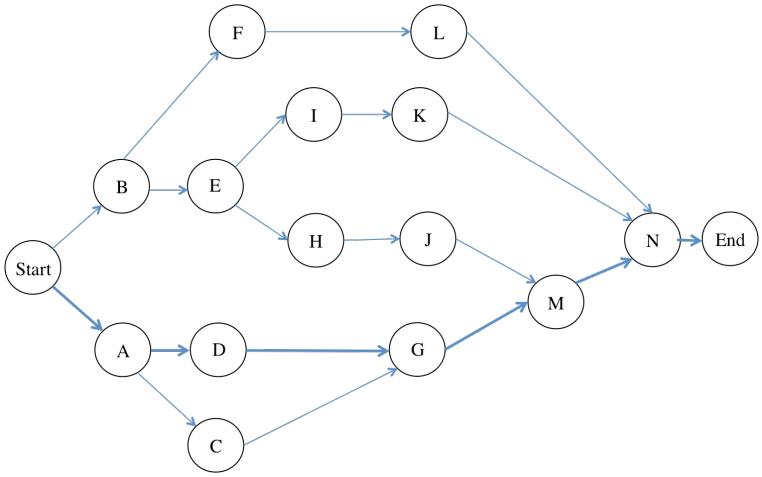


Figure 4.1: Case Study Network

In the last row of Table 4.1, the total project times and costs are indicated for the different modes. For example, if all tasks are performed in Mode 1, the project can be finished in 32 time units with a cost of 198.

The resource (cost) constraint scheduling problem is to find the minimum project time such that the total project cost does not exceed a prescribed limit (equivalence is permitted). The cost constraint in this example is set to $N_r = 236$, which is equal to the cost when all tasks are executed in Mode 2.

Table 4.1: Resource requirements and corresponding durations for three modes

Task	Mode 1 (Regular)		Mode 2 (Fast)		Mode 3 (Extreme)	
	Duration	R1	Duration	R1	Duration	R1
A	8	20	6	24	5	26
B	5	14	4	16	3	19
C	4	15	3	18	2	24
D	8	18	6	22	5	23
E	4	12	3	15	2	19
F	5	16	4	18	3	21
G	6	20	5	22	4	24
H	6	15	4	20	3	24
I	7	15	5	19	4	21
J	5	8	4	9	3	11
K	5	8	4	9	3	10
L	7	15	5	19	4	21
M	6	12	5	13	4	15
N	4	10	3	12	2	13
Total	**32**	**198**	**25**	**236**	**20**	**271**

According to Storn and Price, DE is based on generating new vectors and selecting the vector that survives to the next generation by applying these four steps (Damak et al., 2009).

Step 1: Population Structure

The initial population is randomly generated with respect to precedence constraints. In this example, a 4-individual population is generated (see Table 4.2). Note that the sequence vector contains positions and not the order of execution. For example, in C_1, Task A should be executed at the 3rd place, and not the 3rd task executed in the first place.

We start by selecting randomly four individuals. Three of them will be used in the mutation step, and the fourth will be the target vector. In this example C_2, C_3, C_4 are selected for mutation, and the target will be $Tg = C_1$.

Table 4.2: Initial population

Tasks	A	B	C	D	E	F	G	H	I	J	K	L	M	N
Sequence/ C_1	3	1	7	4	5	2	9	11	6	12	8	10	13	14
Mode/ C_1	1	2	1	2	3	1	2	1	1	3	3	2	1	1
Sequence/ C_2	1	2	3	5	4	11	7	8	6	9	13	12	10	14
Mode/ C_2	3	2	2	1	3	2	3	2	2	3	1	2	3	2
Sequence/ C_3	1	5	3	2	7	6	4	9	11	10	13	8	12	14
Mode/ C_3	3	1	1	2	3	1	1	2	2	2	1	3	3	3
Sequence/ C_4	1	2	10	8	5	3	12	9	6	11	7	4	13	14
Mode/ C_4	1	3	1	3	2	2	3	2	3	1	2	3	2	3

Step 2: Mutation

Two parents C_4 and $C_3 - C_2$ are used to create a child M by using the following equation:

$$M = C_4 + A \cdot R \cdot (C_3 - C_2) \tag{1}$$

A represents a positive number, which controls the evolution rate (should be chosen greater than 1). In this example $A = 1.5$ was chosen. R is a 2 by 14 matrix having uniformly distributed random values between 0 and 1.

Table 4.3: Random Numbers for Sequences and Modes

$rand_1$ seq.	0.23	0.24	0.03	0.14	0.73	0.99	0.16	0.13	0.75	0.43	0.78	0.57	0.24	0.20
$rand_1$ mode	0.08	0.92	0.82	0.01	0.30	0.55	0.92	0.09	0.94	0.77	0.40	0.24	0.49	0.01

Table 4.3 presents the assumed random numbers for each sequence and mode per task. Using Eq. (1), we calculate the mutants for the sequences and modes and the results are combined in a mutant vector and shown in Table 4.4.

For example, the sequence and mode mutant of Task C is:

$$M_{C,S} = 10 + 1.5 * 0.03 * (3 - 3) = 10$$

$$M_{C,m} = 1 + 1.5 * 0.82 * (2 - 5) = -0.23$$

Table 4.4: Mutant Vector

Sequence/ M	1	3.08	10	7.37	8.28	-4.42	11.28	9.19	11.62	11.64	7	0.58	13.7	14
Mode/ M	1	1.62	-0.23	3.01	2	1.17	0.24	2	3	-0.15	2	3.36	2	3.01

Step 3: Crossover

The idea of the crossover is to create a trial vector Tr from the target vector Tg and the mutant vector by crossing them over based on the following procedure:

$$Tr_{i,j} = \begin{cases} M_{i,j} & r_{i,j} \le C_{r,j} \\ Tg_{i,j} & \text{otherwise} \end{cases} \quad (2)$$

Equation (2) defines how the *i,j* component of the trial vector should be calculated after crossover for $i = 1,2$ (sequence/mode) and *j* = A,….., N (tasks). In case the randomly generated $r_{i,j}$ number is less than the prescribed $C_{r,j}$ crossover factor then the mutated element is copied into the (*i,j*) element of the trial vector. Otherwise the corresponding element of the target vector is used.

We start this step by repeating the target vector in Table 4.5, then we perform the crossover and update the target vector using properties of the mutated individual. The random numbers $r_{i,j}$ generated are shown in Table 4.6. The crossover factors are assumed to be $C_{r,1} = 0.2$ for the sequences and $C_{r,2} = 0.1$ for the modes. The crossed-over trial vector is shown in Table 4.7.

Table 4.5: Target Vector

Sequence/ T_g	3	1	7	4	5	2	9	11	6	12	8	10	13	14
Mode/ T_g	1	2	1	2	3	1	2	1	1	3	3	2	1	1

Table 4.6: New Random Numbers for Sequences and Modes

r_1 sequence	0.32	**0.14**	**0.03**	0.97	0.36	0.90	0.24	0.62	0.33	0.27	0.24	0.68	0.62	0.83
r_1 mode	0.23	0.52	0.94	**0.05**	0.59	0.25	0.82	0.43	0.93	**0.03**	0.74	0.95	**0.06**	0.36

Table 4.7: Trial Vector (not arranged)

Sequence/ T_r	3	3.08	10	4	5	2	9	11	6	12	8	10	13	14
Mode/ T_r	1	2	1	3.01	3	1	2	1	1	-0.15	3	2	2	1

Finally, the trial vector will be arranged. First, the sequence vector is manipulated using the precedence relationship and ascending values (this one is the hardest part). Secondly, the mode vector is updated.

We start with the latter because it is easier. The elements of the mode vector are rounded down to the nearest integer. Exceptionally, values less than 1 are adjusted to 1, and the values that exceed the maximal number of modes are converted to 3. Thus,

$$3.01 \rightarrow 3$$
$$-0.15 \rightarrow 1$$
$$2 \rightarrow 2$$

Next, the new sequence has to be created. After the Start node, Task A and Task B compete for execution (see network graph). Since Task A has 3 and Task B has a value of 3.08, Task A gets the first position ($3 < 3.08$).

Next, the successors of Task A (= Task C and Task D) and the remaning Task B compete for second place. Their values respectively are: 10, 4, 3.08. Thus, Task B wins ($3.08 < 4 < 10$).

Next, the successors of Task B (= Task E and Task F) and the remaning Task C, Task D compete for third place. Their values respectively are: 5, 2, 10, 4. Thus, Task F wins ($2 < 4 < 5 < 10$).

Next, the successors of Task F (= Task K and Task L) and the remaning Task C, Task D, Task E compete for fourth place. Their values respectively are: 8, 10, 10, 4, 5. Thus, Task

D wins (4 < 5 < 8 < 10 = 10). Here, a very important rule has to be mentioned: at this point, Task K could not have been chosen since it requires Task I to be finished. It can be chosen only after both Task F and Task I are finished (see network graph). Luckily, we don't have to deal with this since Task K does not have the minimum value. Damak et al. don't handle this kind of problem for two reasons: 1) a computer process can be run after one of its predecessor finished but not all and 2) they might not encounter this kind of problem due to the network they use

Next, the successor of Task D (= Task G) and the remaining Task C, Task E, Task K (still cannot be chosen), Task L compete for fifth place. Their values respectively are: 9, 10, 5, 8, 10. Thus, Task E wins.

Next, the successor of Task E (= Task H and Task I) and the remaining Task C, Task K, Task L compete for sixth place. Their values respectively are: 11, 6, 10, 8, 10. Thus, Task I wins. After this step Task K can be chosen if it has the minimal value since ALL of its predecessors are completed (Task F and Task I).

These rules have to be followed until we reach the end to get the sequence vector of the new solution. The result is shown in Table 4.8. The first six positions that have been calculated are highlighted (Task A, Task B, Task F, Task D, Task E, Task I).

Table 4.8: Trial Vector (arranged)

Tasks	A	B	C	D	E	F	G	H	I	J	K	L	M	N
Sequence/ *S*	**1**	**2**	8	**4**	**5**	**3**	9	11	**6**	12	7	10	13	14
Mode/ *S*	1	2	1	**3**	3	1	2	1	1	**1**	3	2	**2**	1

Step 4: Selection

Selection will be performed between the target vector (Table 4.5) and the trial vector (Table 4.8). This is where we keep the better individual and discard the worse one, i.e. the new generation will be more evolved. Note the target was (C_2) originally the second individual in the population. Thus, this place has to be updated.

$$C_2^{new} = \begin{cases} Tr & f(Tr) \leq f(Tg) \\ Tg & otherwise \end{cases} \quad (3)$$

Equation 3 defines how we should select the individual from the current generation to create the next, hopefully better, generation. To do this we use the fitness function which is the sum of the total makespan for the given individual and a penalty.

$$\text{Fitness} = C_{max} + \text{Penalty} \quad (4)$$

Here C_{max} is calculated by solving the network for the modes of the given individual, i.e. C_{max} is the length of the critical path. The makespan of *Tr* and *Tg* are

$$Tr_{max} = 27$$

$$Tg_{max} = 29$$

We are seeking the minimum completion time. Thus simply, if the fitness value of the target vector *Tg* is greater than the fitness of the new solution vector *Tr*, we set the second individual of the new population to be *Tr,* i.e. $C_2^{new} = Tr$ and we keep all the other individuals from the previous generation and start all over from Step 1. In the opposite case the target vector is kept and no modification occurs to the population. At this point, *Tr* has lower fitness, but we still need to calculate the penalties too.

The better solution is not necessarily feasible due to the resource constraint, so we need to somehow filter them. This is done by the penalty function. Table 4.9 shows the costs of *Tr* and *Tg* respectively.

$$Penalty = \partial . \max\left(\frac{Resource\ Consumed}{Resource\ Threshold} - 1{,}0\right) \quad (5)$$

Table 4.9: Costs of Trial and Target Vector

Tasks	A	B	C	D	E	F	G	H	I	J	K	L	M	N	
Sequence/ S	1	2	8	4	5	3	9	11	6	12	7	10	13	14	
Mode/ S	1	2	1	3	3	1	2	1	1	1	3	2	2	1	
Cost/ S	20	16	15	23	19	16	22	15	15	8	10	19	13	10	$\sum 221$
Sequence/ T_g	3	1	7	4	5	2	9	11	6	12	8	10	13	14	
Mode/ T_g	1	2	1	2	3	1	2	1	1	3	3	2	1	1	
Cost/ T_g	20	16	15	22	19	16	22	15	15	11	10	19	12	10	$\sum 222$

The penalty function has one purpose, namely to indicate if a given mode and sequence vector would produce an infeasible solution, i.e. if the threshold levels of the resources are violated. Since Evolutionary Algorithms cannot directly handle nonlinear constraints of the optimization variables, hence a common approach is to increase the fitness value in case of infeasibility by a penalty function (as in Damak's paper).

Here, the penalty function is chosen in proportion of the level of infeasibility, i.e. the more we violate the resource thresholds the greater penalty we get. Thus, the penalty function in our case can be considered.

$$Penalty = \partial \cdot (\text{"Level Of Infeasibility"})$$

If ∂ is increased, the penalty becomes a (death) penalty i.e. it is more likely that the given mode and sequence vector will not survive to the next generation. On the other hand, if ∂ is decreased then it is more likely for an infeasible solution to survive the next generation. The trade-off has to be found, since we want to eliminate infeasibility from the population, hoping to find the GLOBAL optimal solution; however, we also want some perturbations (degenerates, infeasible solutions) to avoid stagnation in a LOCAL optimum.

Using $\partial = 35.2$ (chosen 10% higher than max cmax), the penalty functions are:

$Penalty_{Tr}$= 35.2 * max$\left[\frac{221}{236} - 1{,}0\right]$= 35.2 * max(-0.0636, 0)= 35.2 * 0 = 0

$Penalty_{Tg}$= 35.2 * max$\left[\frac{222}{236} - 1{,}0\right]$= 35.2 * max(-0.0593, 0)= 35.2 * 0 = 0

When the penalty = 0 it means that the solution is feasible. (Clearly, both 221 and 222 costs are less than the prescribed limit of 236). Thus, the overall fitness are:

$$f(Tr) = 27 + 0 \qquad f(Tg) = 29 + 0$$

This means that $f(Tr) \leq f(Tg)$, i.e. $C_2^{new} = Tr$

The new generations are therefore shown in Table 4.10.

Table 4.10: Second generation population

Tasks	A	B	C	D	E	F	G	H	I	J	K	L	M	N
Sequence/ C_1^{new}	1	2	8	4	5	3	9	11	6	12	7	10	13	14
Mode/ C_1^{new}	1	2	1	3	3	1	2	1	1	1	3	2	2	1
Sequence/ C_2^{new}	9	1	11	10	2	3	12	5	4	8	7	6	13	14
Mode/ C_2^{new}	1	1	1	2	3	1	2	1	1	3	3	2	1	1
Sequence/ C_3^{new}	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Mode/ C_3^{new}	3	1	1	2	3	1	1	2	2	2	1	3	3	3
Sequence/ C_4^{new}	2	1	6	5	4	3	12	8	7	11	10	9	13	14
Mode/ C_4^{new}	3	2	2	1	3	2	3	2	2	3	1	2	3	2

4.2 Multi Mode Single Non Renewable (MMSNR) Resource Constrained Project Scheduling Problem

The single resource project scheduling problem is a simple optimization problem with one resource (R1 or Cost) involved. Using Table 4.1, the Cost constraint is set to 236; thus, the sum of all cost usage should not exceed this limit (equivalence is permitted). By using differential evolution algorithm, the optimal solution will indicate the best mode for each activity to be executed and determine the optimal total cost and total project time. The objective of resource constraint project scheduling problems is to find the precedence and resource feasible completion times for all activities minimize the makespan of the project.

The computer program consists of two levels (Appendix A). At low-level the critical path computation is performed using topological sort with finding the longest path in the directed acyclic graph (project network) and backpropagation for slack time calculation. At high-level the program performs the steps of the DEA algorithm which is repeated for every generation until the stopping criterion is met. The connection between the two level is the fitness function where the low-level function is called for a given individual in the population. Constraints enforce the presedence constraints between activities and constraint limits for each resource type k and each time instant t that the resource demand of the activities which are currently processed does not exceeed the capacity (Weglarz, 1999).

Finding Optimal Solution:

```
Project Time = 20.00
Penalty = 0.00
Fitness = 20.00

Activity    Mode vector       Time  Cost  Critical

      A               3        5.0  26.0     yes
      B               3        4.0  16.0     yes
      C               1        4.0  15.0      no
      D               3        5.0  23.0     yes
      E               2        2.0  19.0     yes
      F               1        5.0  16.0      no
      G               3        4.0  24.0     yes
      H               2        4.0  20.0     yes
      I               1        7.0  15.0     yes
      J               2        4.0   9.0     yes
      K               1        5.0   8.0     yes
      L               1        7.0  15.0      no
      M               3        4.0  15.0     yes
      N               3        2.0  13.0     yes

Total cost: 234.0
Total project time: 20.0
```

Verification of Results:

Total cost: 26+19+15+23+15+16+24+20+15+9+8+15+15+13 = 234

Project time (A-D-G-M-N): 5+5+4+4+2 = 20

Project time (B-E-H-J-M-N): 5+5+4+4+2 = 20

Project time (B-E-I-K-N): 4+2+7+5+2 = 20

Table 4.11: Mode Selected for Each Resource According to the Optimal Solution (highlighted)

Task	Mode 1 (Regular)		Mode 2 (Fast)		Mode 3 (Extreme)	
	Duration	R1	Duration	R1	Duration	R1
A	8	20	6	24	5	26
B	5	14	4	16	3	19
C	4	15	3	18	2	24
D	8	18	6	22	5	23
E	4	12	3	15	2	19
F	5	16	4	18	3	21
G	6	20	5	22	4	24
H	6	15	4	20	3	24
I	7	15	5	19	4	21
J	5	8	4	9	3	11
K	5	8	4	9	3	10
L	7	15	5	19	4	21
M	6	12	5	13	4	15
N	4	10	3	12	2	13
Total	**32**	**198**	**25**	**236**	**20**	**271**

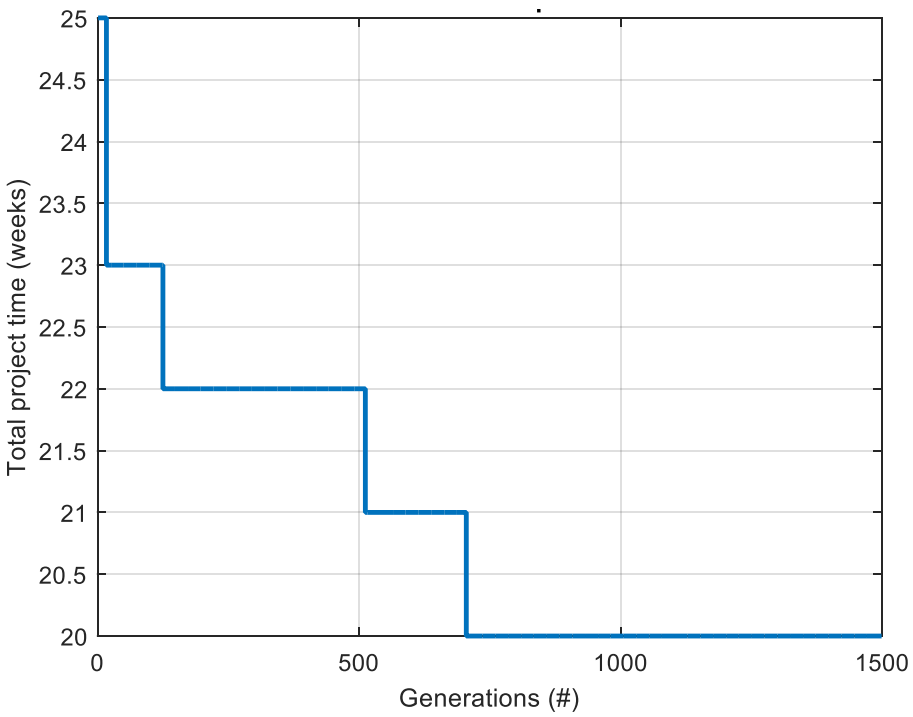


Figure 4.2: Evolution of Feasible Optimal Solution

4.3 Multi Mode Multiple NonRenewable (MMMNR) Resource Constrained Project Scheduling Problem

Three resource types are assumed: R1, R2 and R3 (for example: cost, work hour, material quantity). For example, Task A needs 24 units of R1, 8 units of R2 and 4 units of R3 if executed in Mode 2. If all tasks are executed in extreme mode (Mode 3), total of 271 of R1, 94 of R2 and 58 of R3 would be needed. The total project time would be 20 units of time (from Table 4.12). The total duration is based on critical path.

Table 4.12: Three Resource Requirements for Three Modes

Task	Mode 1 (Regular)				Mode 2 (Fast)				Mode 3 (Extreme)			
	Duration	R1	R2	R3	Duration	R1	R2	R3	Duration	R1	R2	R3
A	8	20	7	3	6	24	8	4	5	26	9	5
B	5	14	5	2	4	16	6	3	3	19	7	4
C	4	15	5	3	3	18	6	3	2	24	8	5
D	8	18	6	3	6	22	8	4	5	23	8	5
E	4	12	4	2	3	15	5	3	2	19	7	4
F	5	16	6	3	4	18	6	3	3	21	7	4
G	6	20	7	3	5	22	8	4	4	24	8	5
H	6	15	5	3	4	20	7	3	3	24	8	5
I	7	15	5	3	5	19	7	3	4	21	7	4
J	5	8	3	2	4	9	3	2	3	11	4	3
K	5	8	3	2	4	9	3	2	3	10	4	3
L	7	15	5	3	5	19	7	3	4	21	7	4
M	6	12	4	2	5	13	5	2	4	15	5	4
N	4	10	4	2	3	12	4	2	2	13	5	3
Total	**32**	**198**	**69**	**36**	**25**	**236**	**83**	**41**	**20**	**271**	**94**	**58**

Constraint Settings:

The following assumptions are made:

- R1 must be LESS or EQUAL to 236
- R2 must be LESS or EQUAL to 83
- R3 must be LESS or EQUAL to 41

These settings are identical to the case when all tasks are executed in Mode 2. Without optimization, the total duration would be 25. The same DEA for single constraint project scheduling program was used but adjusted to include the additional resources R2, R3. The new modes/durations have been added to the program (R2, R3). The penalty function now takes into consideration all resources by computing the penalty function for each resource and then summing it (Appendix B). If all penalties are zero then the feasible mode sequence is found by taking into consideration all resources and limitations.

The extension of multi modes requires the update of the TimeTable and ResourceTable variables according to the specified values. Moreover, in each fitness function call the resource consumption of each activity for each resource has to be tracked. This is done by selecting the corresponding values of the ResourceTable to the actual mode vector. After this, we can compute the total resource consumption/resource type over the project (three values in our case). Dividing them by the corresponding resource thresholds and subtracting 1 from them would give <= 0 for feasible allocation and >0 for infeasible allocation. Clearly the sum of these results will be <= 0 if ALL resource allocations are feasible and >0 if AT LEAST ONE of them are not feasible.

This will be the new penalty function:

$$Penalty = \sum_{i=1}^{N} Penalty_i$$

In our case it is:

$$Penalty = \partial \cdot \max\left(\frac{\text{Total R1 consumption}}{\text{R1 threshold level}} - 1; 0\right) + \partial \cdot \max\left(\frac{\text{Total R2 consumption}}{\text{R2 threshold level}} - 1; 0\right) + \partial \cdot \max\left(\frac{\text{Total R3 consumption}}{\text{R3 threshold level}} - 1; 0\right)$$

Finding Optimal Solution:

```
Total Project Time = 24.00
Penalty = 0.00
Fitness = 24.00

Activity    Mode  Time    R1     R2     R3   Critical    Sequence

    A       2      6.0  24.0    8.0    4.0     yes          1.
    B       1      5.0  14.0    5.0    2.0     yes          2.
    C       1      4.0  15.0    5.0    3.0      no          3.
    D       2      6.0  22.0    8.0    4.0     yes          4.
    E       1      4.0  12.0    4.0    2.0     yes          5.
    F       2      4.0  18.0    6.0    3.0      no          7.
    G       2      5.0  22.0    8.0    4.0     yes          8.
    H       2      4.0  20.0    7.0    3.0     yes         10.
    I       1      7.0  15.0    5.0    3.0     yes          6.
    J       2      4.0   9.0    3.0    2.0     yes         11.
    K       1      5.0   8.0    3.0    2.0     yes         12.
    L       1      7.0  15.0    5.0    3.0      no          9.
    M       3      4.0  15.0    5.0    4.0     yes         13.
    N       2      3.0  12.0    4.0    2.0     yes         14.

                     Threshold   >=   Consumption
Resource Type  1:   236.00                221.00
Resource Type  2:   83.00                 76.00
Resource Type  3:   41.00                 41.00
```

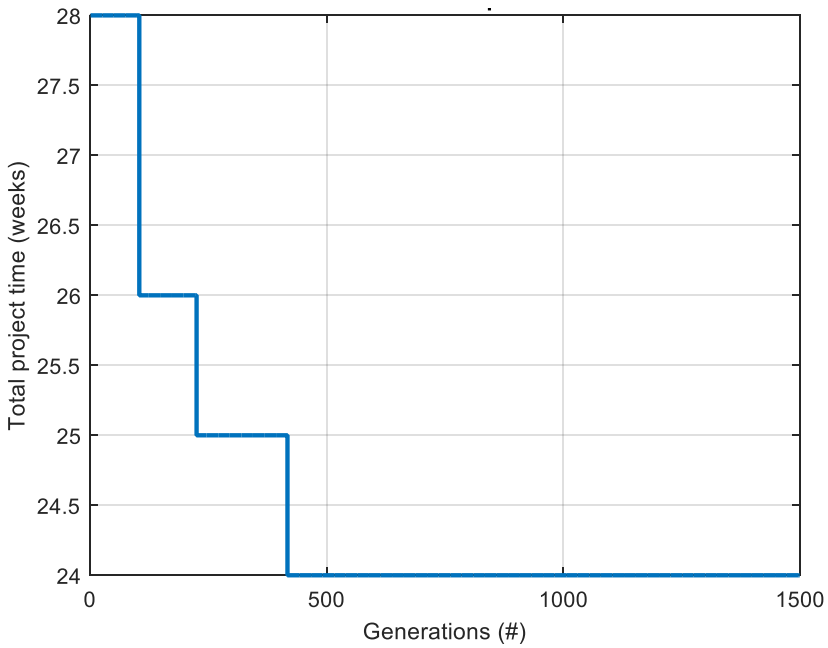


Figure 4.3: Evolution of Feasible Optimal Solution

The optimal solution is 24 weeks (the optimization decreased the TPT 1 week compared to the 25 week Mode 2 solution). The optimal TPT increased by 4 weeks compared to the single resource solution.

Table 4.13: Mode Selected for Each Resource According to the Optimal Solution (highlighted)

Task	Mode 1 (Regular)				Mode 2 (Fast)				Mode 3 (Extreme)			
	Duration	R1	R2	R3	Duration	R1	R2	R3	Duration	R1	R2	R3
A	8	20	7	3	6	24	8	4	5	26	9	5
B	5	14	5	2	4	16	6	3	3	19	7	4
C	4	15	5	3	3	18	6	3	2	24	8	5
D	8	18	6	3	6	22	8	4	5	23	8	5
E	4	12	4	2	3	15	5	3	2	19	7	4
F	5	16	6	3	4	18	6	3	3	21	7	4
G	6	20	7	3	5	22	8	4	4	24	8	5
H	6	15	5	3	4	20	7	3	3	24	8	5
I	7	15	5	3	5	19	7	3	4	21	7	4
J	5	8	3	2	4	9	3	2	3	11	4	3
K	5	8	3	2	4	9	3	2	3	10	4	3
L	7	15	5	3	5	19	7	3	4	21	7	4
M	6	12	4	2	5	13	5	2	4	15	5	4
N	4	10	4	2	3	12	4	2	2	13	5	3
Total	**32**	**198**	**69**	**36**	**25**	**236**	**83**	**41**	**20**	**271**	**94**	**58**

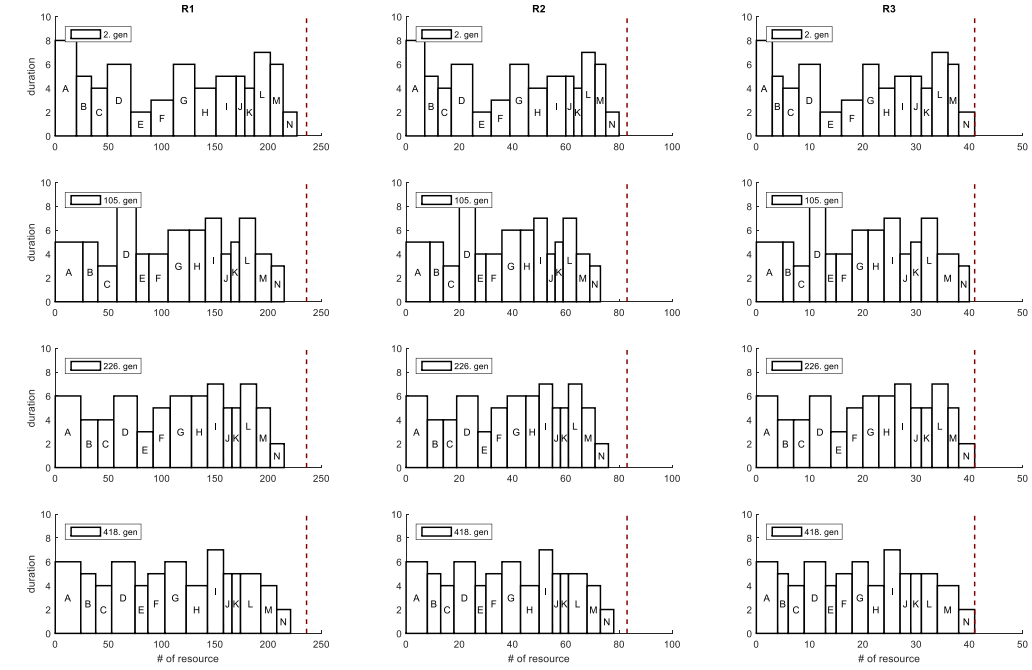
Verification of Results:

R1: 24+14+15+22+12+16+20+20+15+8+8+19+15+13 = 221

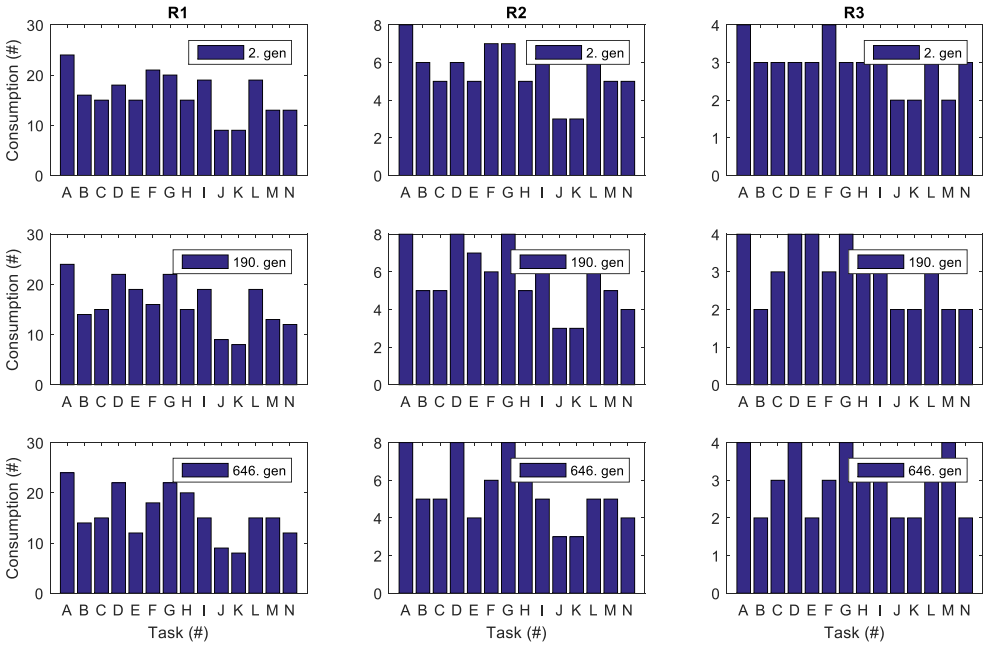
R2: 8+5+5+8+4+6+7+7+5+3+3+7+5+5 = 76

R3: 4+2+3+4+2+3+3+3+3+2+2+3+4+3 = 41

Figure 4.4 presents two graphs to show the evolution of the resource allocation in function of the generation number. The columns in both graphs represent the resource types. In the first graph (A) the total resource consumption with the corresponding threshold levels and the evolution of the optimal duration of the activities can be seen. The second graph (B) shows the individual consumption of each activity for each resource over the generations.



(A)



(B)

Figure 4.4: Resource Allocations

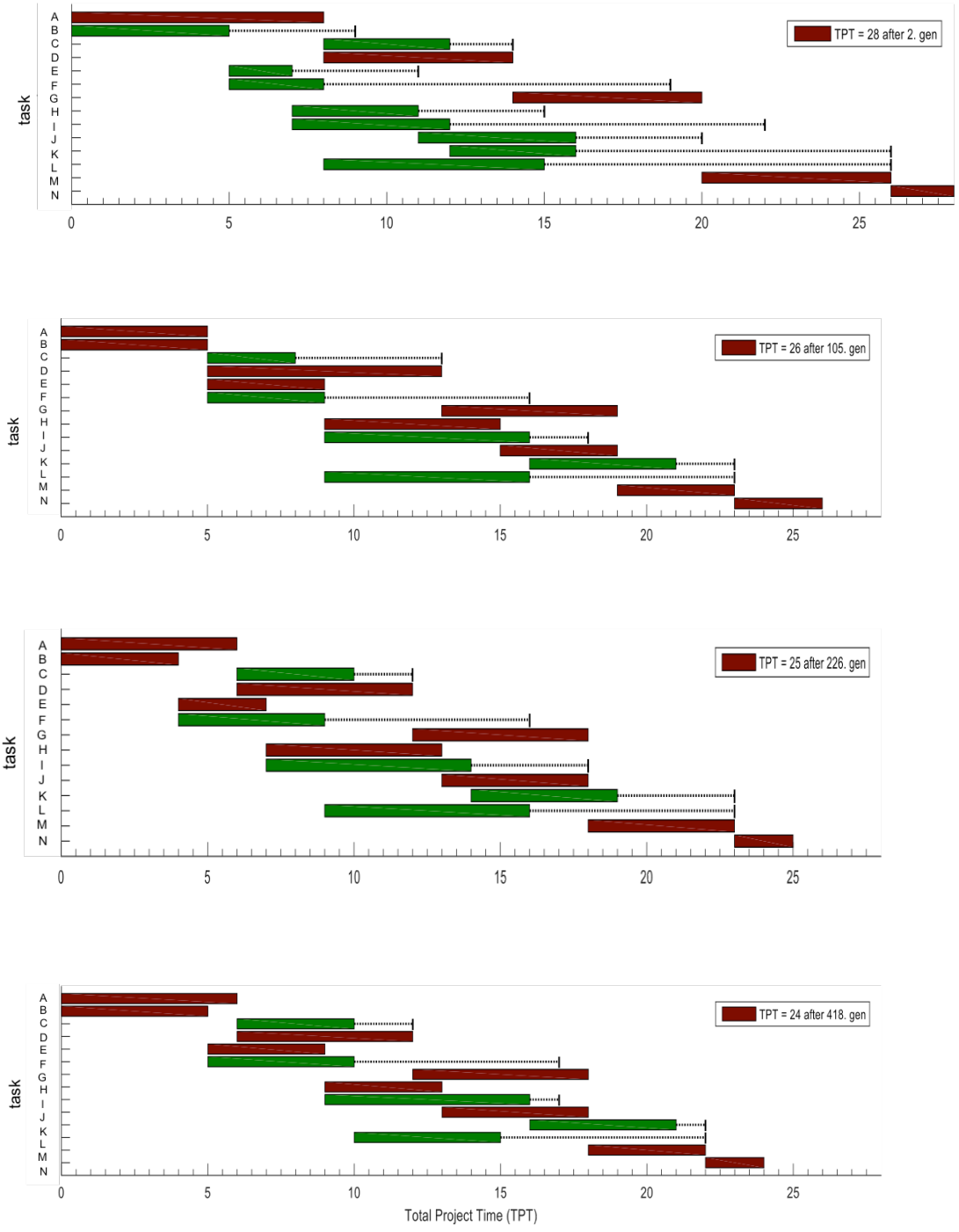


Figure 4.5: Gantt Charts

Figure 4.5 shows the evolution of the optimal scheduling over the generations by using Gantt charts. The red activities are critical; the green ones have slack time and they can be shifted along the dotted lines until reaching their end such that the optimal solution is not violated. The optimal scheduling is 24 weeks long (after the 646th generation).

Figure 4.6 shows the final optimal network structure with the highlighted critical path(s) and the modes. This corresponds to the solution given in Table 4.13.

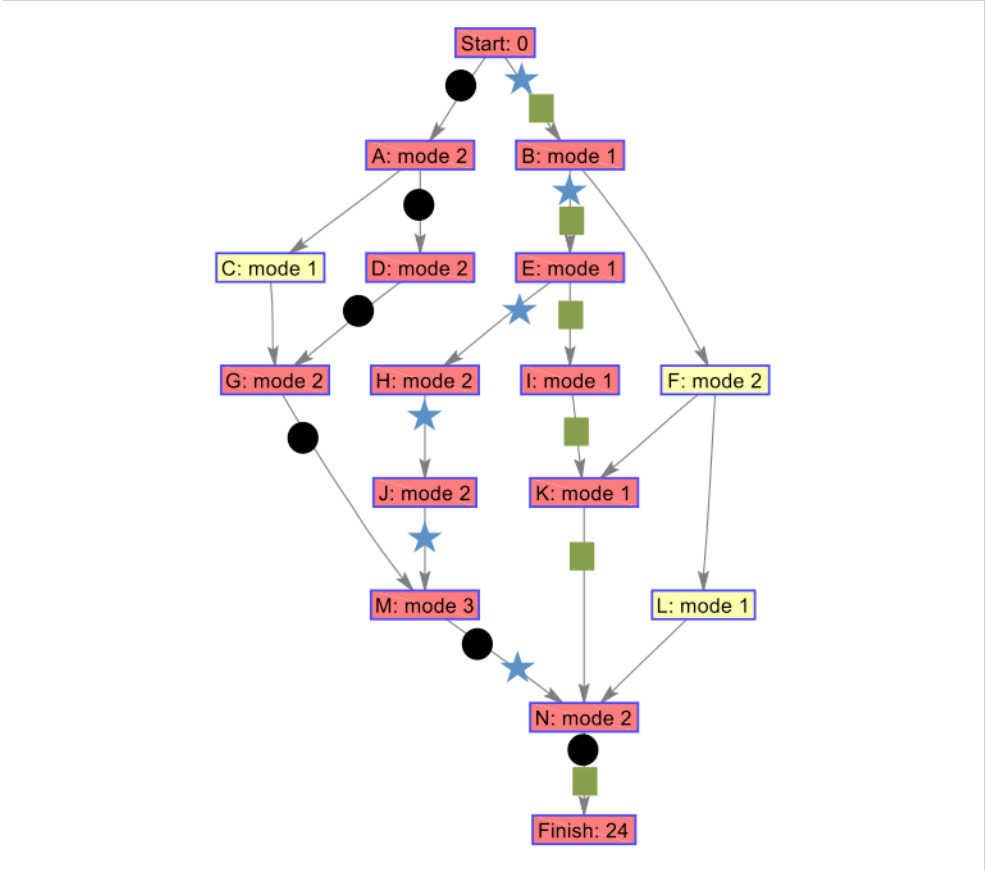


Figure 4.6: Optimal Network Structure

4.4 Multi Mode Single Non Renewable (MMSNR) Resource Constrained Project Scheduling Problem with Weekly Constraint

The same network in Figure 4.1 and same resource requirements in Table 4.1 are applied here. A new constraint is added by introducing an extra penalty function (implemented in xpenalty) which is added to the already existing one. The new function first computes the weekly resource needs of the different modes (resource need/duration). According to the current mode vector, a matrix similar to the Gantt Chart can be created which keeps track of the weekly resource consumption of each activity between its starting and ending time. Since activities can be executed in parallel (satisfying precence constraints), the total consumption of a given week can be obtained by adding up the weekly resource consumption for ALL active tasks in that week. Doing this for ALL weeks, we can check whether there is a week with higher consumption than the given weekly threshold level. If so, penalty is introduced proportionally to the number of weeks that are violated in this level (Appendix C).

The total resource threshold level is still set at 236. Additionally to this constraint, a weekly threshold level is defined. It is assumed that in each week (separately) the consumption of R1 cannot exceed this threshold.

In the next example the weekly threshold constraint is assumed to be 15. Note that the project can be done in 20 weeks below 236 total R1 usage. This means that 236/20 = 11.8 is the average consumption of each week. The problem is to find the minimum makepsan of the project that satisfies BOTH constraints (weekly and total threshold).

Finding Optimal Solution:

```
Constraints settings:
 - weekly constraints set = 15
 - maximum resource consumption allowed is 236.0

Generations:
# 100
…
#2500
Elapsed time is 48.920122 seconds.

Optimal solution:
-----------------
Total Project Time = 22.00
Penalty1 = 0.00
Penalty2 = 0.00
Fitness  = 22.00

Activity    Mode  Time   R1   Critical    Sequence

    A       3      5.0  26.0     yes          1.
    B       3      3.0  19.0     yes          4.
    C       1      4.0  15.0      no          2.
    D       2      6.0  22.0     yes          3.
    E       1      4.0  12.0     yes          6.
    F       2      4.0  18.0      no          7.
    G       2      5.0  22.0     yes          5.
    H       1      6.0  15.0     yes          8.
    I       1      7.0  15.0      no         10.
    J       3      3.0  11.0     yes         11.
    K       3      3.0  10.0      no         13.
    L       1      7.0  15.0      no          9.
    M       3      4.0  15.0     yes         12.
    N       3      2.0  13.0     yes         14.

                   Threshold   >=   Consumption
Resource Type  1:  236.00               228.00
Weekly Threshold:  15.00
```

Table 4.14: Weekly Resource Needs

Activity	A	B	C	D	E	F	G	H	I	J	K	L	M	N	SUM
Week1	5.2	6.3	0	0	0	0	0	0	0	0	0	0	0	0	**11.5**
Week2	5.2	6.3	0	0	0	0	0	0	0	0	0	0	0	0	**11.5**
Week3	5.2	6.3	0	0	0	0	0	0	0	0	0	0	0	0	**11.5**
Week4	5.2	0	0	0	3	4.5	0	0	0	0	0	0	0	0	**12.7**
Week5	5.2	0	0	0	3	4.5	0	0	0	0	0	0	0	0	**12.7**
Week6	0	0	3.75	3.66	3	4.5	0	0	0	0	0	0	0	0	**14.9**
Week7	0	0	3.75	3.66	3	4.5	0	0	0	0	0	0	0	0	**14.9**
Week8	0	0	3.75	3.66	0	0	0	2.5	2.1	0	0	2.1	0	0	**14.2**
Week9	0	0	3.75	3.66	0	0	0	2.5	2.1	0	0	2.1	0	0	**14.2**
Week10	0	0	0	3.66	0	0	0	2.5	2.1	0	0	2.1	0	0	**10.4**
Week11	0	0	0	3.66	0	0	0	2.5	2.1	0	0	2.1	0	0	**10.4**
Week12	0	0	0	0	0	0	4.4	2.5	2.1	0	0	2.1	0	0	**11.2**
Week13	0	0	0	0	0	0	4.4	2.5	2.1	0	0	2.1	0	0	**11.2**
Week14	0	0	0	0	0	0	4.4	0	2.1	3.6	0	2.1	0	0	**12.3**
Week15	0	0	0	0	0	0	4.4	0	0	3.6	3.3	0	0	0	**11.4**
Week16	0	0	0	0	0	0	4.4	0	0	3.6	3.3	0	0	0	**11.4**
Week17	0	0	0	0	0	0	0	0	0	0	3.3	0	3.75	0	**7.1**
Week18	0	0	0	0	0	0	0	0	0	0	0	0	3.75	0	**3.75**
Week19	0	0	0	0	0	0	0	0	0	0	0	0	3.75	0	**3.75**
Week20	0	0	0	0	0	0	0	0	0	0	0	0	3.75	0	**3.75**
Week21	0	0	0	0	0	0	0	0	0	0	0	0	0	6.5	**6.5**
Week22	0	0	0	0	0	0	0	0	0	0	0	0	0	6.5	**6.5**

Each task through out the weeks should add up to its total values. For example:

Task A: 5.2/ week for five weeks which is 5.2 * 5 = 26

Another example:

Task J needs 3.6/ week for three weeks, which is 3.6 * 3 = 10.8 in total. However, Table 4.12 shows that in mode 3 Task J consumes 11 resources. This is still valid since it is actually 3.666666666666666...*3 = 10.99999999999999.... = 11.

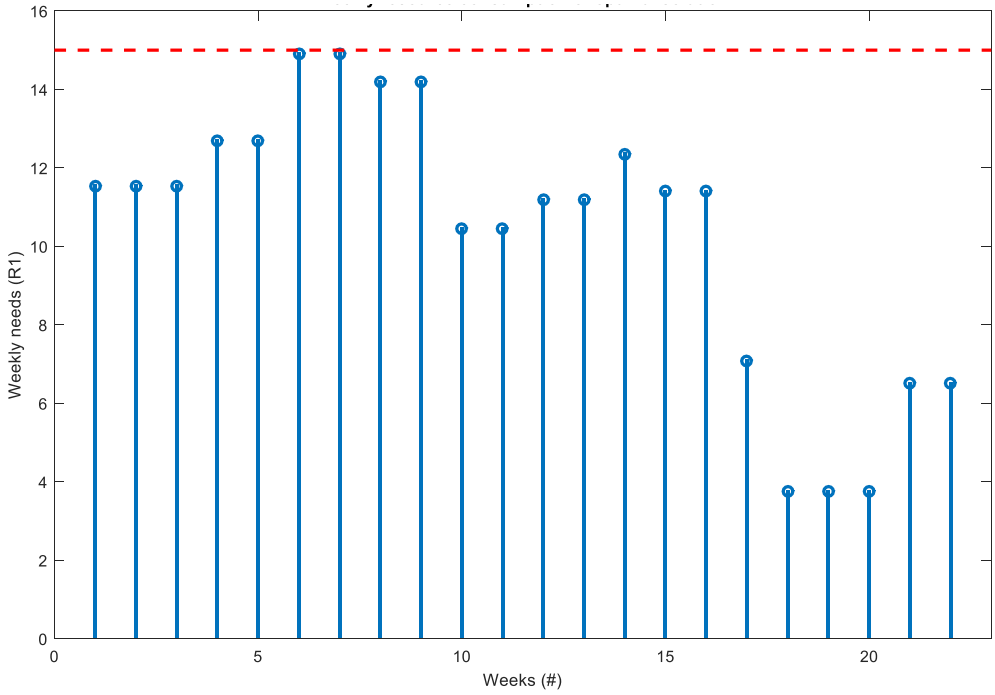


Figure 4.7: Weekly Resource Consumption of Optimal Solution

Figure 4.7 shows the total weekly consumption for the optimal solution (tpt = 22 weeks) along with the threshold level. It can be seen that around the 6-7th week, the consumption was the highest but still not over the limit; thus, the optimal solution is indeed feasible.

Next, the weekly threshold level is varied between 10 and 20. The results are shown below.

```
Constraints settings:
 - weekly constraints set between 10.0 and 20.0
 - maximum resource consumption allowed is 236.0

DEA optimization started...
OPT  1/11 ... Problem is infeasible!
OPT  2/11 ... Problem is infeasible!
OPT  3/11 ... Problem is infeasible!
OPT  4/11 ... Solution found!
OPT  5/11 ... Solution found!
OPT  6/11 ... Solution found!
OPT  7/11 ... Solution found!
OPT  8/11 ... Solution found!
OPT  9/11 ... Solution found!
OPT 10/11 ... Solution found!
OPT 11/11 ... Solution found!
```

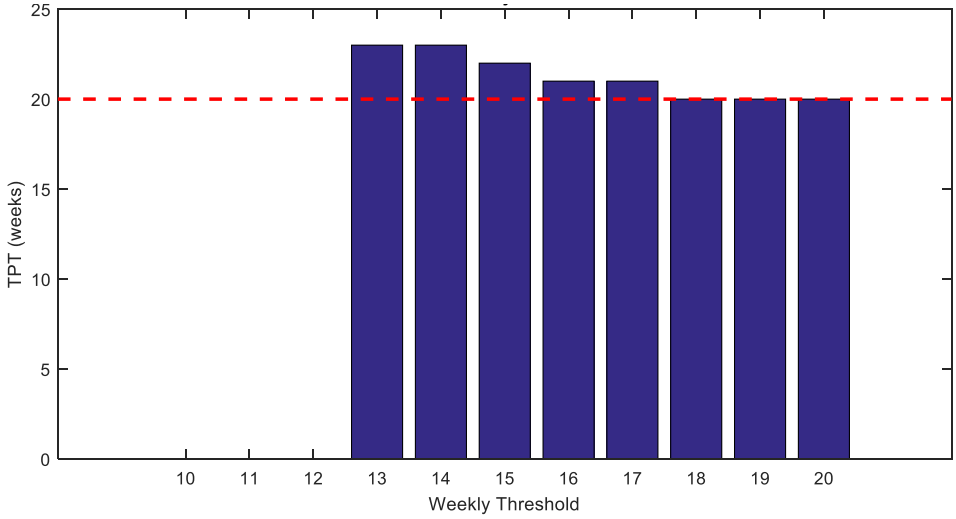


Figure 4.8: Total Project Time of the Weekly Threshold

Figure 4.8 shows the total project time in function of the weekly threshold level along with the optimal project time without weekly consraints (single resource, TPT = 20 weeks, red dashed line). It can be seen that if the threshold is below 13 than no feasible solution exist. Between 13 and 17 the original optimal 20 week project time is increased due to the stronger constraints imposed on the project. Above the 18 weekly threshold level, we get the same 20 week solution.

For 236 total R1, the TPT are shown above in function of the weekly thresholds:

weekly threshold <= 12	no feasible solution exist
weekly threshold = 13	TPT = 23
weekly threshold = 14	TPT = 23
weekly threshold = 15	TPT = 22
weekly threshold = 16	TPT = 21
weekly threshold = 17	TPT = 21
weekly threshold >= 18	TPT = 20

4.5 Multi Mode Multiple Non Renewable (MMMNR) Resource Constrained Project Scheduling Problem with Weekly Constraints

The same network in Figure 4.1 and the same resource requirements in Table 4.12 are applied here. Three new constraints were added by introducing an extra penalty function (implemented in xpenalty) which is added to the already existing one. The new function first computes the weekly resource needs of the different modes (resource need/duration) AND different resources. According to the current mode vector, a matrix similar to the Gantt Chart can be created, this is called a Summary Table, which keeps track of the weekly resource consumption of each activity between its starting and ending time. Since activities can be executed in parallel (satisfying precence constraints), the total consumption of a given week and given resource can be obtained by adding up the weekly resource consumption for ALL active tasks in that week and for that resource. Doing this for ALL weeks we can check whether there is a week with higher consumption than the given weekly threshold level for a given resource. If so, penalty is introduced proportionally to the number of weeks that are violated this level. FOR ALL three resources, we can obtain three different penalties. The goal is to make them zero, i.e. no penalty is needed for the optimal solution. This means that the sum of the penalyties is also needed to be zero (Appendix D).

The total resource threshold level is still set at [236, 83, 41]. In addition to this constraint, three weekly threshold levels are defined. It is assumed that in each week (separately) the consumption of R1, R2 and R3 cannot exceed these thresholds: 15, 5 and 3, respectively.

In the next example the weekly threshold constraints are assumed to be [15, 5, 3]. Note that the project can be done in 24 weeks (see Scenario 2) with 226 total R1 usage and 76 total R2 usage and 41 total R3 usage. This means that 226/24 = 9.42, 76/24 = 3.2, 41/24 = 1.71 is the average consumption of the resources in each week. The problem is to find the

minimum makepsan of the project that satisfies BOTH constraints types (weekly and total threshold) for ALL THREE resources.

Finding Optimal Solution:

DEA optimization started...

Generations:
100
…
#1500
Elapsed time is 34.458197 seconds.

Optimal solution:

Total Project Time = 24.00
Penalty1 = 0.00
Penalty2 = 0.00
Fitness = 24.00

Activity	Mode	Time	R1	R2	R3	Critical	Sequence
A	3	5.0	26.0	9.0	5.0	yes	1.
B	1	5.0	14.0	5.0	2.0	yes	3.
C	1	4.0	15.0	5.0	3.0	no	7.
D	1	8.0	18.0	6.0	3.0	yes	2.
E	1	4.0	12.0	4.0	2.0	yes	5.
F	1	5.0	16.0	6.0	3.0	no	4.
G	3	4.0	24.0	8.0	5.0	yes	12.
H	2	4.0	20.0	7.0	3.0	yes	8.
I	1	7.0	15.0	5.0	3.0	no	6.
J	2	4.0	9.0	3.0	2.0	yes	9.
K	1	5.0	8.0	3.0	2.0	no	11.
L	1	7.0	15.0	5.0	3.0	no	10.
M	2	5.0	13.0	5.0	2.0	yes	13.
N	3	2.0	13.0	5.0	3.0	yes	14.

	Threshold >=	Consumption
Resource Type 1:	236.00	218.00
Resource Type 2:	83.00	76.00
Resource Type 3:	41.00	41.00

Table 4.15: Weekly Resource Needs of R1

Activity	A	B	C	D	E	F	G	H	I	J	K	L	M	N	SUM
Week 1	5.2	2.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**8.0**
Week 2	5.2	2.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**8.0**
Week 3	5.2	2.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**8.0**
Week 4	5.2	2.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**8.0**
Week 5	5.2	2.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**8.0**
Week 6	0.0	0.0	3.8	2.3	3.0	3.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**12.2**
Week 7	0.0	0.0	3.8	2.3	3.0	3.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**12.2**
Week 8	0.0	0.0	3.8	2.3	3.0	3.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**12.2**
Week 9	0.0	0.0	3.8	2.3	3.0	3.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**12.2**
Week 10	0.0	0.0	0.0	2.3	0.0	3.2	0.0	5.0	2.1	0.0	0.0	0.0	0.0	0.0	**12.6**
Week 11	0.0	0.0	0.0	2.3	0.0	0.0	0.0	5.0	2.1	0.0	0.0	2.1	0.0	0.0	**11.5**
Week 12	0.0	0.0	0.0	2.3	0.0	0.0	0.0	5.0	2.1	0.0	0.0	2.1	0.0	0.0	**11.5**
Week 13	0.0	0.0	0.0	2.3	0.0	0.0	0.0	5.0	2.1	0.0	0.0	2.1	0.0	0.0	**11.5**
Week 14	0.0	0.0	0.0	0.0	0.0	0.0	6.0	0.0	2.1	2.3	0.0	2.1	0.0	0.0	**12.5**
Week 15	0.0	0.0	0.0	0.0	0.0	0.0	6.0	0.0	2.1	2.3	0.0	2.1	0.0	0.0	**12.5**
Week 16	0.0	0.0	0.0	0.0	0.0	0.0	6.0	0.0	2.1	2.3	0.0	2.1	0.0	0.0	**12.5**
Week 17	0.0	0.0	0.0	0.0	0.0	0.0	6.0	0.0	0.0	2.3	1.6	2.1	0.0	0.0	**12.0**
Week 18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.6	0.0	2.6	0.0	**4.2**
Week 19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.6	0.0	2.6	0.0	**4.2**
Week 20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.6	0.0	2.6	0.0	**4.2**
Week 21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.6	0.0	2.6	0.0	**4.2**
Week 22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.6	0.0	**2.6**
Week 23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.5	**6.5**
Week 24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.5	**6.5**

Table 4.16: Weekly Resource Needs of R2

Activity	A	B	C	D	E	F	G	H	I	J	K	L	M	N	SUM
Week 1	1.8	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**2.8**
Week 2	1.8	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**2.8**
Week 3	1.8	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**2.8**
Week 4	1.8	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**2.8**
Week 5	1.8	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**2.8**
Week 6	0.0	0.0	1.3	0.8	1.0	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**4.2**
Week 7	0.0	0.0	1.3	0.8	1.0	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**4.2**
Week 8	0.0	0.0	1.3	0.8	1.0	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**4.2**
Week 9	0.0	0.0	1.3	0.8	1.0	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**4.2**
Week 10	0.0	0.0	0.0	0.8	0.0	1.2	0.0	1.8	0.7	0.0	0.0	0.0	0.0	0.0	**4.4**
Week 11	0.0	0.0	0.0	0.8	0.0	0.0	0.0	1.8	0.7	0.0	0.0	0.7	0.0	0.0	**3.9**
Week 12	0.0	0.0	0.0	0.8	0.0	0.0	0.0	1.8	0.7	0.0	0.0	0.7	0.0	0.0	**3.9**
Week 13	0.0	0.0	0.0	0.8	0.0	0.0	0.0	1.8	0.7	0.0	0.0	0.7	0.0	0.0	**3.9**
Week 14	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.7	0.8	0.0	0.7	0.0	0.0	**4.2**
Week 15	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.7	0.8	0.0	0.7	0.0	0.0	**4.2**
Week 16	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.7	0.8	0.0	0.7	0.0	0.0	**4.2**
Week 17	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.8	0.6	0.7	0.0	0.0	**4.1**
Week 18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.0	1.0	0.0	**1.6**
Week 19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.0	1.0	0.0	**1.6**
Week 20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.0	1.0	0.0	**1.6**
Week 21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.0	1.0	0.0	**1.6**
Week 22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	**1.0**
Week 23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.5	**2.5**
Week 24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.5	**2.5**

Table 4.17: Weekly Resource Needs of R3

Activity	A	B	C	D	E	F	G	H	I	J	K	L	M	N	SUM
Week 1	1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**1.4**
Week 2	1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**1.4**
Week 3	1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**1.4**
Week 4	1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**1.4**
Week 5	1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**1.4**
Week 6	0.0	0.0	0.8	0.4	0.5	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**2.2**
Week 7	0.0	0.0	0.8	0.4	0.5	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**2.2**
Week 8	0.0	0.0	0.8	0.4	0.5	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**2.2**
Week 9	0.0	0.0	0.8	0.4	0.5	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**2.2**
Week 10	0.0	0.0	0.0	0.4	0.0	0.6	0.0	0.8	0.4	0.0	0.0	0.0	0.0	0.0	**2.2**
Week 11	0.0	0.0	0.0	0.4	0.0	0.0	0.0	0.8	0.4	0.0	0.0	0.4	0.0	0.0	**2.0**
Week 12	0.0	0.0	0.0	0.4	0.0	0.0	0.0	0.8	0.4	0.0	0.0	0.4	0.0	0.0	**2.0**
Week 13	0.0	0.0	0.0	0.4	0.0	0.0	0.0	0.8	0.4	0.0	0.0	0.4	0.0	0.0	**2.0**
Week 14	0.0	0.0	0.0	0.0	0.0	0.0	1.3	0.0	0.4	0.5	0.0	0.4	0.0	0.0	**2.6**
Week 15	0.0	0.0	0.0	0.0	0.0	0.0	1.3	0.0	0.4	0.5	0.0	0.4	0.0	0.0	**2.6**
Week 16	0.0	0.0	0.0	0.0	0.0	0.0	1.3	0.0	0.4	0.5	0.0	0.4	0.0	0.0	**2.6**
Week 17	0.0	0.0	0.0	0.0	0.0	0.0	1.3	0.0	0.0	0.5	0.4	0.4	0.0	0.0	**2.6**
Week 18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.4	0.0	**0.8**
Week 19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.4	0.0	**0.8**
Week 20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.4	0.0	**0.8**
Week 21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.4	0.0	**0.8**
Week 22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	**0.4**
Week 23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.5	**1.5**

Each task for each resource through out the weeks should add up to its total values.

For example:

Task A (R1 resource): 5.2/week for five weeks which is 5.2 * 5 = 26

The following figures show the weekly consumption of R1, R2 and R3 resources along with the corresponding threshold levels. It can be seen that all weekly contraints are satisfied. The

Weekly Threshold >= Consumption (weekly maximum)

Resource R1: 15.00 12.59

Resource R2: 5.00 4.41

Resource R3: 3.00 2.61

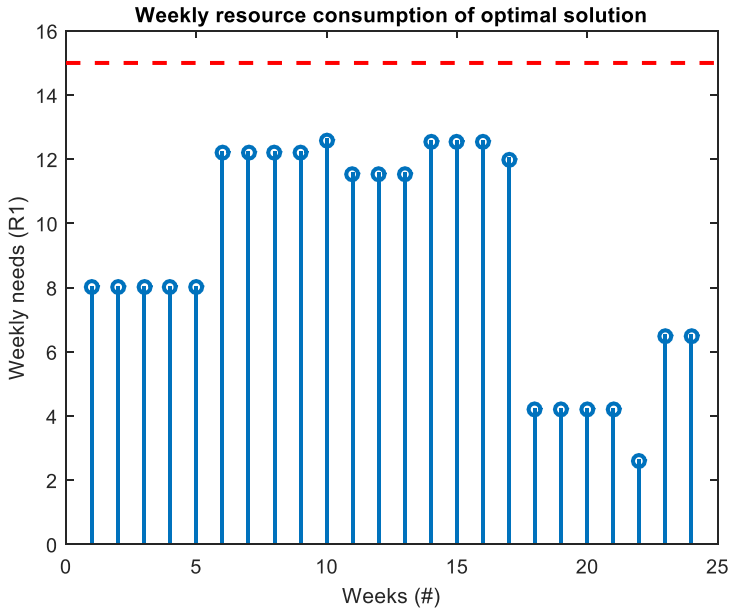


Figure 4.9: Weekly Resource Consumption of Optimal Solution of R1

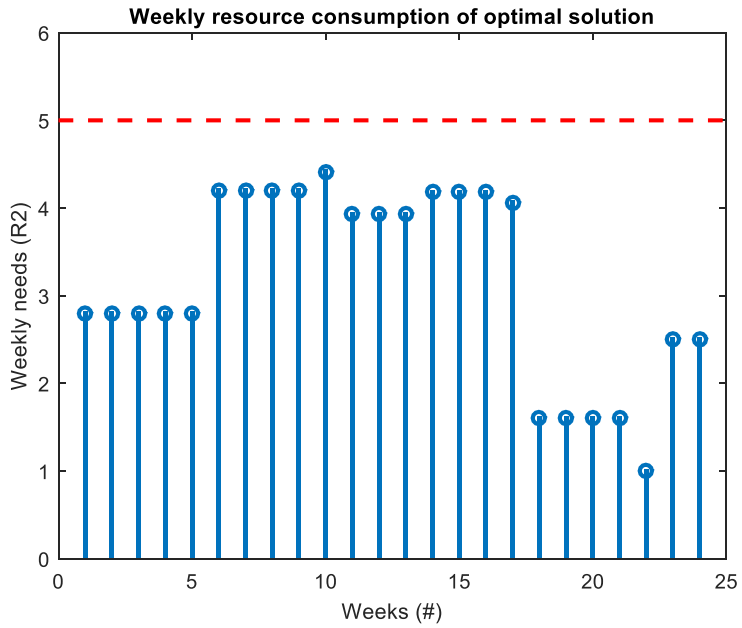


Figure 4.10: Weekly Resource Consumption of Optimal Solution of R2

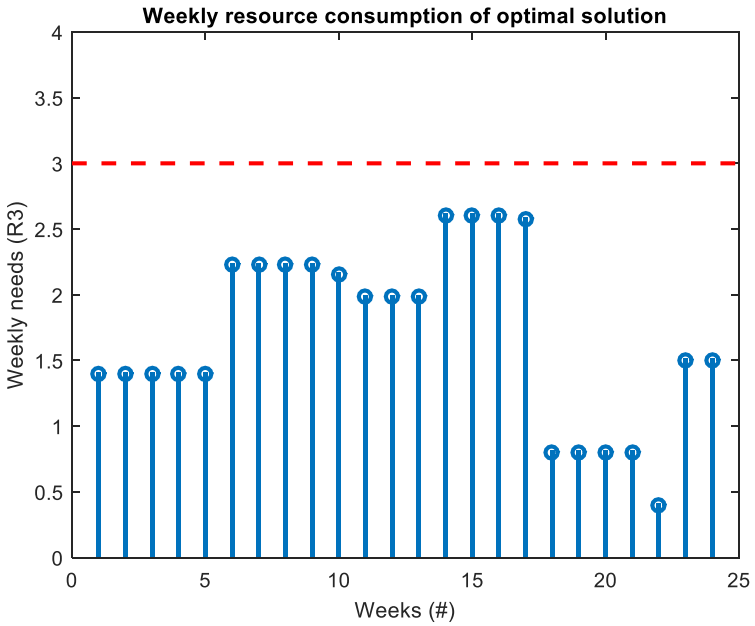


Figure 4.11: Weekly Resource Consumption of Optimal Solution of R3

Figure 4.9, 4.10, 4.11 show the total weekly consumption for the optimal solution (TPTt = 24 weeks) along with the threshold levels. It can be seen that between the 5-15th week, the consumption was the highest but still not over the limit; thus, the optimal solution is indeed feasible.

4.6 Cost and Time Contingencies for Multi Mode Single Non Renewable (MMSNR) Resource Constrained Project Scheduling Problem

The same network in Figure 4.1 and the same resource requirements in Table 4.1 are applied here. In the initial DEA optimization after 1500 generations the optimal TPT= 20 week using 236 threshold level using Table 4.1. Assuming in the 9th week (vertical red dashed line) the budget drops by approximatly 10%. This means that task1 (A), task2 (B), task3 (C), task5 (E) and task6 (F) is already finished. Tasks D, H, I are in progress. The red boxes are the critical tasks, the green boxes are the non-ciritcal tasks scheduled "as early as possible". After the 9th week a new DEA optimizes the remaining network (without tasks A,B,C,E,F) to minimizes the additional delay due to budget drop.

The main difficulty here is building up a second project network, which is considered to be started just right after the budget drop appeared by preserving the very last „states" of the original network. This means, that already finished tasks has to be eliminated and the durations of the activities in progress have to be decreased by the „already finsihed amount of job". These „in-progress" actvities are connected with a NEW Start mode and once the new network is completed the DEA algorithm can be invoked. The algorithm can be done by the following steps (Appendix E):

1) Acquire current completion of the project by the result of the first DEA
2) Find "in-progress" and "already-finsihed" activities. The latter nodes should be eliminated.
3) Insert a new START node
4) Since some nodes might have been eliminated, hence renumber the new project.
5) Update the following: edge list of the network (G), TimeTable.
6) Compute resource consumed until budget drop and subtract it from the threshold.
7) Finally, update the remaining ResourceTable according to the new network.

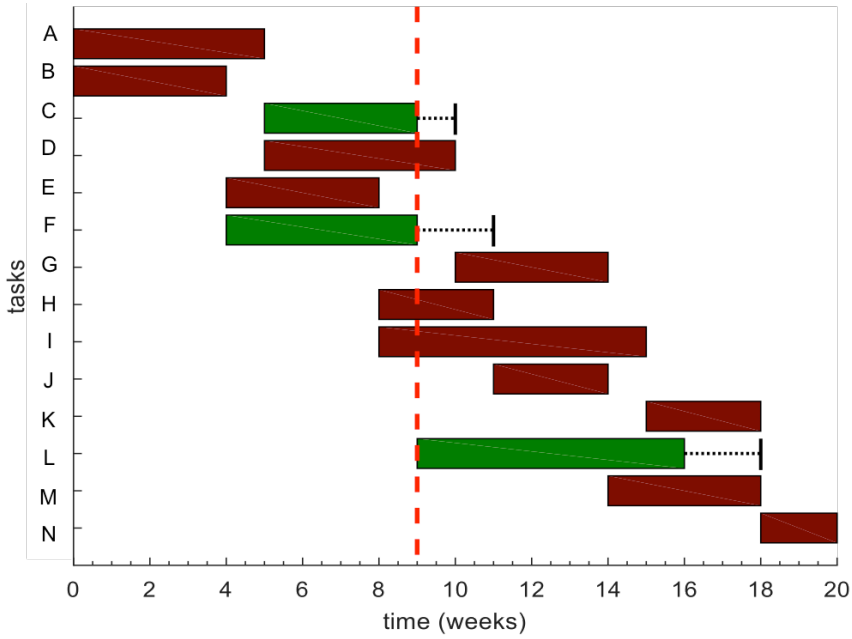


Figure 4.12: Initial DEA with Showing 9th Week Completed and Uncompleted Tasks

The cost of 236 is needed to finish in 20 weeks. At the 9th week approximately 113.5 was the consumption already. The remaining budget is then 236 - 113.5 = 122.5 approximately. This 122.5 is needed to finish every task according to the first plan. However, if the budget 122.5 is dropped (lowered) by 10 units of the 9th week budget it will actually be 10/122.5 = 8.16% budget drop.

The 112.5 cost falls between the pure mode 1 and mode 2 completion time (17 and 14 weeks respectively). Still, the re-optimized network can produce 13 weeks (with 112.5 cost) for the remaining tasks. Thus, the total project time due to budget drop would be 9 + 13 = 22 weeks, which is 2 weeks more than the original 20 weeks.

By using MC simulation we can investigate the overall delay in function of the budget drop. Noticing that almost all activities became critical.

Finding Optimal Solution for the Remaining Network After 9th Week:

```
Mode  Cost  TPT
  1   106.5 17.0
  2   117.5 14.0
  3   127.5 11.0

New cost threshold: 112.5

DEA optimization
----------------
Progress: 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%

Optimal solution
----------------
Cost threshold = 112.5
Total project cost = 112.5
Total project time = 13.0
Penalty = 0.00

Activity    Mode        Time  Cost  Critical

    D       3            1.0   4.6     yes
    G       1            6.0  20.0     yes
    H       3            2.0  16.0     yes
    I       2            6.0  12.9     yes
    J       1            5.0   8.0     yes
    K       1            5.0   8.0     yes
    L       1            7.0  15.0      no
    M       3            4.0  15.0     yes
    N       3            2.0  13.0     yes
```

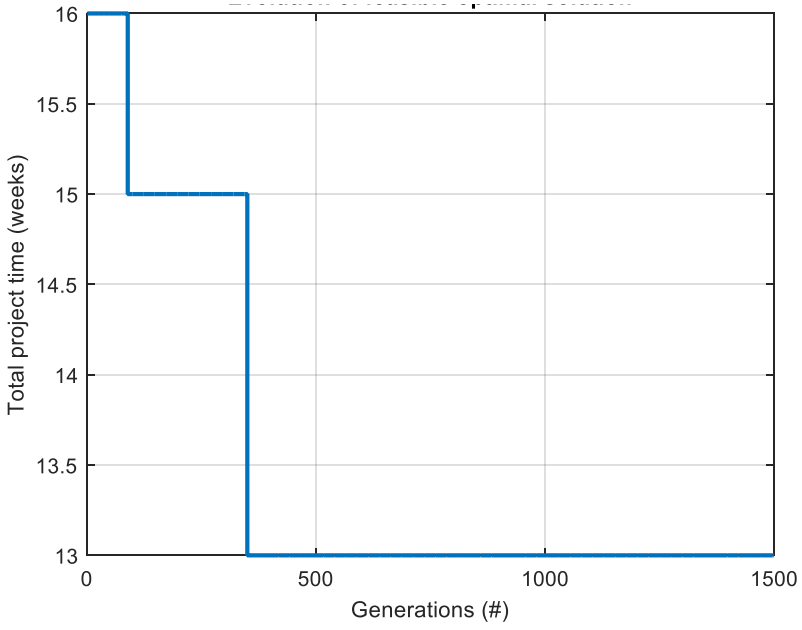


Figure 4.13: Evolution of Feasible Optimal Solution

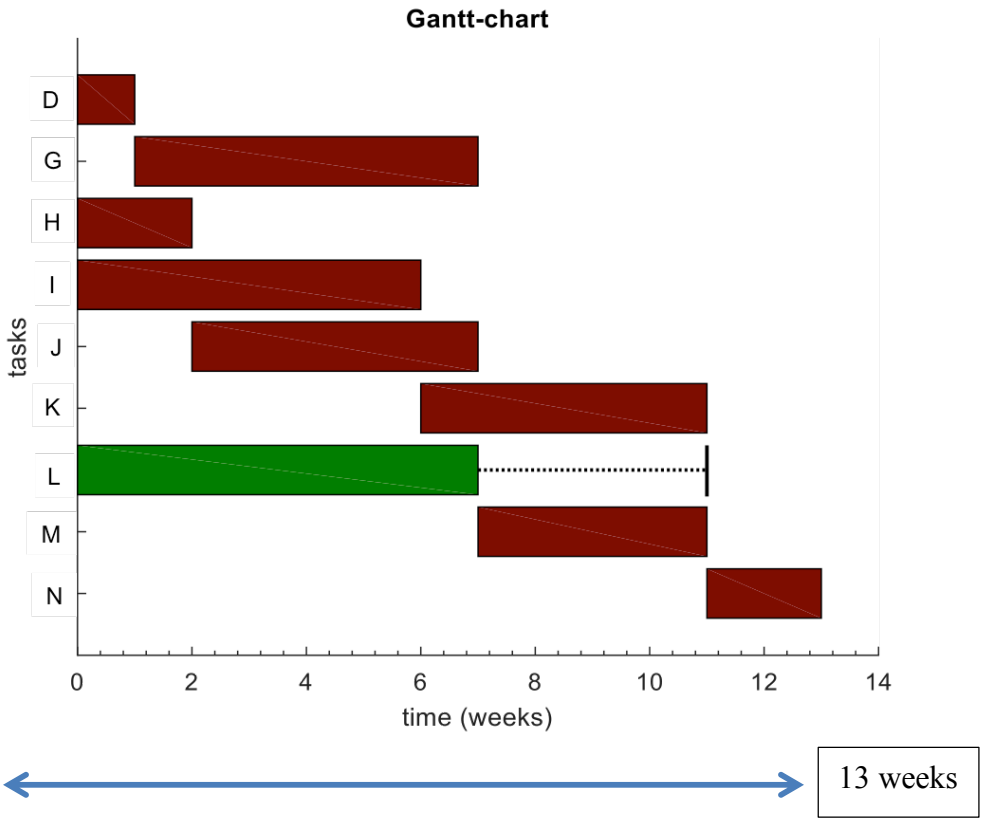


Figure 4.14: DEA for Remaining Network

Table 4.18: Cost of Resources Each Week

Activity	Cost of Regular	Cost of Fast	Cost of Extreme
D	18/8= 2.25	22/6= 3.66	23/5= 4.6
G	20/6= 3.33	22/5= 4.4	24/4= 6
H	15/6= 2.5	20/4= 5	24/3= 8
I	15/7= 2.14	19/5= 3.8	21/4= 5.25
J	8/5= 1.6	9/4= 2.25	11/3= 3.66
K	8/5= 1.6	9/4= 2.25	10/3= 3.33
L	15/7= 2.14	19/5= 3.8	21/4= 5.25
M	12/6= 2	13/5= 2.6	15/4= 3.75
N	10/4= 2.5	12/3= 4	13/2= 6.5

D	Task4	(1 week)	mode3	cost = 4.6 * 1 week = 4.6
G	Task7	(6 week)	mode1	cost = 3.33 * 6 week = 20
H	Task8	(2 week)	mode3	cost = 8 * 2 week = 16
I	Task9	(6 week)	mode2	cost = 3.8 * 6 week = 12.9
J	Task10	(5 week)	mode1	cost = 1.6 * 5 week = 8
K	Task11	(5 week)	mode1	cost = 1.6 * 5 week = 8

L	Task12	(7 week)	mode1	cost = 2.14 * 7 week = 15
M	Task13	(4 week)	mode3	cost = 3.75 * 4 week = 15
N	Task14	(2 week)	mode3	cost = 6.5 * 2 week = 13

Total cost = 4.6 +20 +16 +12.9 + 8 + 8 + 15 + 15 + 13 = 112.5

TPT = 9 + 13 weeks = 22 weeks in total

4.7 Cost and Time Contingencies for Multi Mode Multiple Non Renewable (MMMNR) Resource Constrained Project Scheduling Problem

The same network in Figure 4.1 and same resource requirements in Table 4.12 are applied here. In the initial DEA optimization after 1500 generations the optimal TPT= 24 week using threshold levels of [236, 83, 41] for R1, R2 and R3, respectively (second scenario).

Finding Optimal Solution Before Budget Drop:

```
Total Project Time = 24.00
Penalty = 0.00
Fitness = 24.00

Activity    Mode   Time R1     R2    R3   Critical    Sequence

    A       3      5.0  26.0   9.0   5.0      no          1.
    B       1      5.0  14.0   5.0   2.0     yes          2.
    C       1      4.0  15.0   5.0   3.0      no          4.
    D       2      6.0  22.0   8.0   4.0      no          5.
    E       1      4.0  12.0   4.0   2.0     yes          3.
    F       1      5.0  16.0   6.0   3.0      no          7.
    G       2      5.0  22.0   8.0   4.0      no          6.
    H       2      4.0  20.0   7.0   3.0     yes          9.
    I       2      5.0  19.0   7.0   3.0      no         11.
    J       2      4.0   9.0   3.0   2.0     yes         10.
    K       1      5.0   8.0   3.0   2.0      no          8.
    L       2      5.0  19.0   7.0   3.0      no         14.
    M       2      5.0  13.0   5.0   2.0     yes         12.
    N       3      2.0  13.0   5.0   3.0     yes         13.

                   Threshold   >=   Consumption
Resource Type  1:  236.00           228.00
Resource Type  2:  83.00            82.00
Resource Type  3:  41.00            41.00
```

Assuming in the 9th week (vertical red dashed line) the budget drops by approximately 10%. This means that task1 (A), task2 (B), task3 (C), and task5 (E) are already finished. Task D and Task F are in progress. The red boxes are the critical tasks, the green boxes are the non-ciritcal tasks scheduled “as early as possible”.

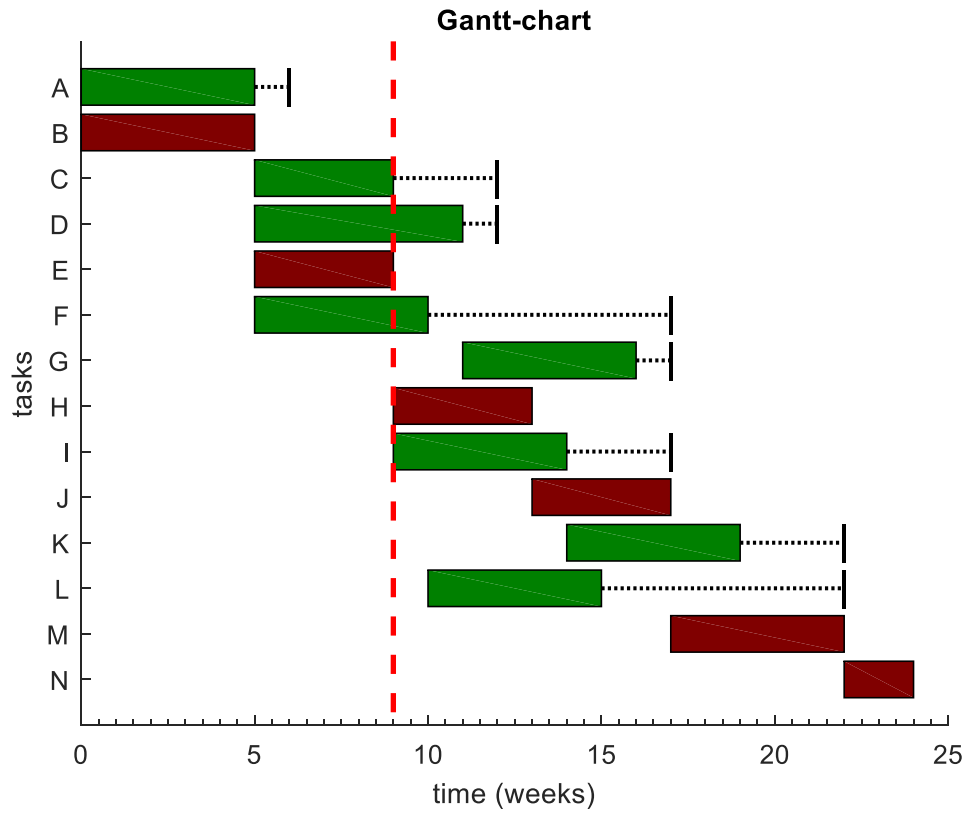


Figure 4.15: Initial DEA with Showing 9th Week Completed and Uncompleted Tasks

After the 9th week a new DEA optimizes the remaining network (without tasks A,B,C,E) to minimizes the additional delay due to budget drop.

The main difficulty here is building up a second project network, which is considered to be started just right after the budget drop appeared by preserving the very last „states" of the original network. This means, that already finished tasks has to be eliminated and the durations of the activities in progress have to be decreased by the „already finsihed amount of job". These „in-progress" actvities are connected with a NEW Start mode and once the new network is completed the DEA algorithm can be invoked. The algorithm can be done by the following steps (Appendix F):

1) Acquire current completion of the project by the result of the first DEA
2) Find "in-progress" and "already-finsihed" activities. The latter nodes should be eliminated.

3) Insert a new START node
4) Since some nodes might have been eliminated, hence renumber the new project.
5) Update the following: edge list of the network (G), TimeTable.
6) Compute resource consumed until budget drop and subtract it from the threshold.
7) Finally, update the remaining ResourceTable according to the new network by using an universal Summary Table which contains every cost details for each task and for each time instance.

R1 <= 236, R2 <= 83, R3 <= 41 are needed to finish in 24 weeks.

At the 9th week the consumption was already:

Table 4.19: Resources After Budget Drop (Until 9th Week)

Task	Completion	Opt. Mode	R1 cost	R2 cost	R3 cost
A	5/5=100%	3	26*100%=26	9*100%=9	5*100%=5
B	5/5=100%	1	14*100%=14	5*100%=5	2*100%=2
C	4/4=100%	1	15*100%=15	5*100%=5	3*100%=3
D	4/6=66.67%	2	22*66.67%=14.67	8*66.67%=5.33	4*66.67%=2.67
E	4/4=100%	1	12*100%=12	4*100%=4	2*100%=2
F	4/5=80%	1	16*80%=12.8	6*80%=4.8	3*80%=2.4
			∑ = 94.47	**∑ = 33.13**	**∑ = 17.07**

Therefore, the remaining resources at the 9th week a:

R= the assumed resource thereshold – the consumed resource

R1 = 236 – 94.47 = 141.53

R2 = 83 – 33.13 = 49.87

R3 = 41 – 17.07 = 23.93

These [141.53,49.87,23.93] resources are needed to finish every task according to the first plan. However, after a 10% budget drop, the new resources become

141.53*0.9 = 127.38 approximately 128

49.87*0.9 = 44.88 approximately 45

23.93*0.9 = 21.54 approximately 22

The pre-calculations show that 128 (R1) resource falls between the pure mode 1 and mode 2 completion time (21 and 17 weeks respectively). 45 (R2) resource falls between the pure mode 1 and mode 2 completion time (21 and 17 weeks respectively). 22 (R3) resource falls between the pure mode 1 and mode 2 completion time (21 and 17 weeks respectively).

Pre-Calculations:

Mode	R1	R2	R3	TPT
1	113.5	39.9	21.9	21.0
2	133.5	47.9	22.9	17.0
3	149.5	51.9	32.9	12.0

Example of R2 (mode 2)= (8*33.33%) + (6*20%) + 8 + 7 + 7 + 3 + 3 + 7 + 5 + 4 = 47.9

Finding Optimal Solution for the Remaining Network After 9th Week:

Pre-Calculations:

Mode	R1	R2	R3	TPT
1	113.5	39.9	21.9	21.0
2	133.5	47.9	22.9	16.0
3	149.5	51.9	32.9	12.0

New R1 threshold = 128.0
New R2 threshold = 45.0
New R3 threshold = 22.0

DEA optimization:

Progress: 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%

Optimal solution:

Total Project Time = 16.00
Penalty = 0.00
Fitness = 16.00

Activity	Mode	Time	R1	R2	R3	Critical	Sequence
D	3	2.0	7.3	2.7	1.3	yes	6.
F	1	1.0	3.2	1.2	0.6	no	2.
G	1	6.0	20.0	7.0	3.0	yes	1.
H	2	4.0	20.0	7.0	3.0	yes	5.
I	1	7.0	15.0	5.0	3.0	no	10.
J	2	4.0	9.0	3.0	2.0	yes	7.

```
 K        1    5.0     8.0     3.0     2.0      no      9.
 L        1    7.0    15.0     5.0     3.0      no      8.
 M        2    5.0    13.0     5.0     2.0     yes      3.
 N        2    3.0    12.0     4.0     2.0     yes      4.

                Threshold  >=  Consumption
Resource Type  1:  128.00          122.53
Resource Type  2:  45.00           42.87
Resource Type  3:  22.00           21.93
```

Still, the re-optimized network can produce 16 weeks with [122.53, 42.87, 21.93] cost for the remaining tasks. Thus, the total project time due to budget drop would be 9 + 16 = 25 weeks, which is only 1 week more than the original optimal 24 weeks.

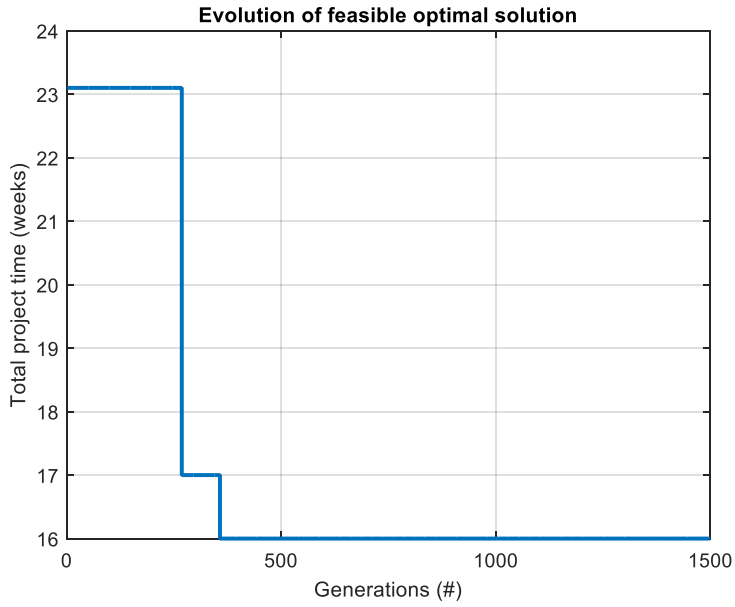


Figure 4.16: Evolution of Feasible Optimal Solution after Budget Drop

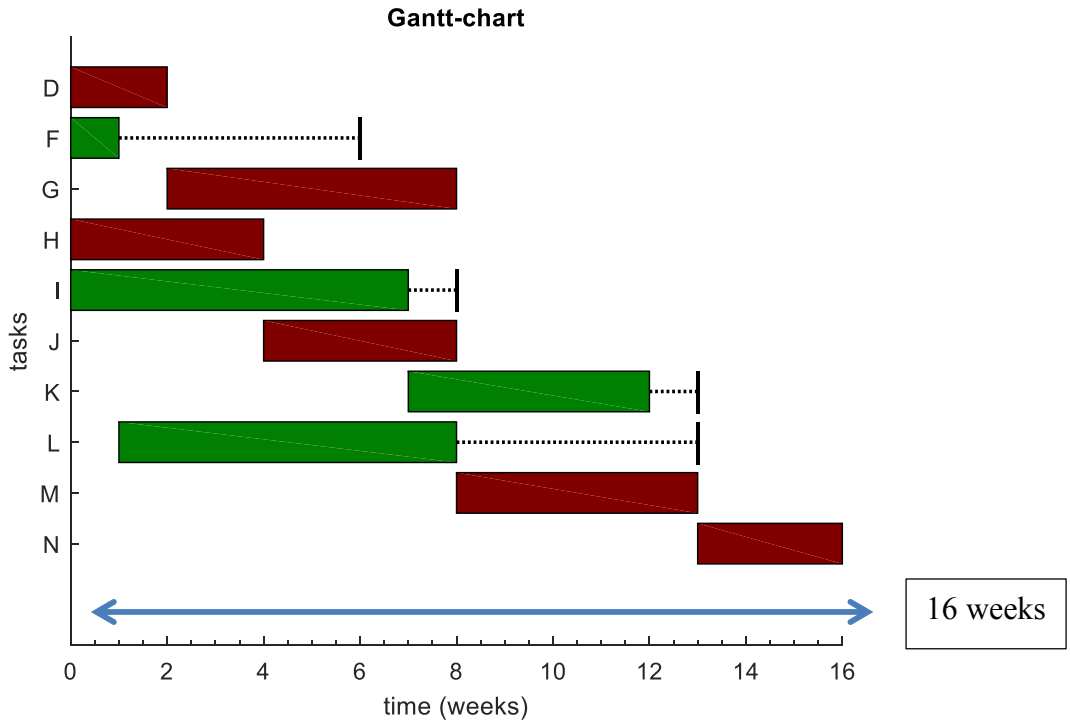


Figure 4.17: DEA for Remaining Network (after Budget Drop)

Table 4.20: Resources After Budget Drop (After 9th Week)

Task	Completion	Optimal Mode	R1 cost	R2 cost	R3 cost
D	2/6=33.3%	3*	22×33.3%=7.33	8×33.3%=2.67	4×33.3%=1.33
F	1/5=20%	1*	16×20%=3.2	6×20%=1.2	3×20%=0.6
G	6/6	1	20	7	3
H	4/4	2	20	7	3
I	7/7	1	15	5	3
J	4/4	2	9	3	2
K	5/5	1	8	3	2
L	7/7	1	15	5	3
M	5/5	2	13	5	2
N	3/3	2	12	4	2
			∑ = 122.53	∑ = 42.87	∑ = 21.93

Task D and Task F were already completed by 66.67% and 80% until the 9th week, so only 33.33% and 20% remain. In Table 4.20, although Task D follows mode 3 after the

budget drop, it is still calculated with mode 2 as indicated in the DEA before the budget drop. It is assumed that if a task is running at the time of the drop, its mode cannot be altered after the drop. The task has to finish in its original mode.

For example, if the task is to produce 4 specific materials in the automotive industry, and already the aluminium panels have been cut, the robots programmed, the workers assigned, etc. these numbers cannot be altered. They can only be changed after everything is finished.

Total resource consumption is:

R1 = 94.47 (before 9th week) + 122.53 (after 9th week) = 217

R2 = 33.13 (before 9th week) + 42.87 (after 9th week) = 76

R3 = 17.07 (before 9th week) + 21.93 (after 9th week) = 39

More Generally:

Completion of Task *i* at time *T*

$$= \begin{cases} 0 \; if \; F_i \; > T \\ 1 \; if \; F_i \; \leq T \end{cases}$$

$\frac{T-S_i}{F_i-S_i}$ if $F_i \; > T$

Example:

Task D ($S_D = 5, F_D = 11$)

T= g when budget drop

Completion Time of Task D= $\frac{T-S_i}{F_i-S_i} = \frac{9-5}{11-5} = \frac{4}{6}$ = 66.7%

CHAPTER 5

BENCHMARK RESULTS & PARETO FRONTIER

5.1 Benchmark Results

The effectiveness of the DEA algorithm is compared to a bechmark problem given in paper *Ant Colony Optimization for Multimode Resource-Constrained Project Scheduling* by Hong Zhang (Journal of Management in Engineering, vol 2., issue 2, pp 150-159, 2012). The benchmark network is from page 155. The actvities are identified as A, B, C, D, E, F, G, H, I, J instead of 1-10 from now on.

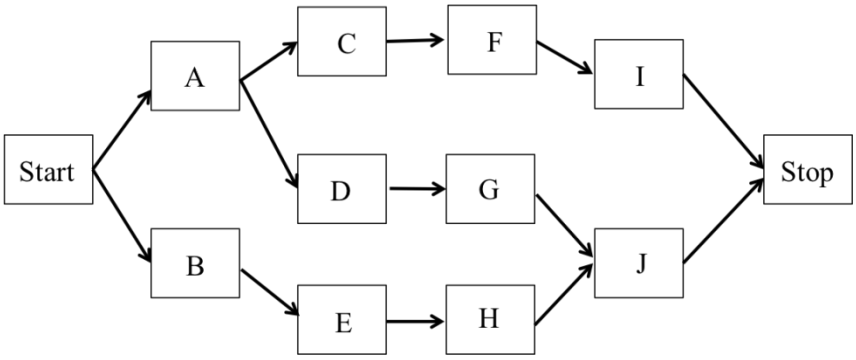


Figure 5.1: Benchmark Network

This benchmark problem uses a single RENEWABLE resource with three modes. NOTE that up to this point only stronger constraints (as NON-RENEWABLE) are imposed on the project networks. Renewable constraints are easier to handle. The resource needs and the corresponding durations are shown in Table 5.1. Notice that activities 4, 6 and 7 (or D, F and G) can only be executed in mode 1 and 2.

Table 5.1: Information About the Three Modes

	Mode 1		Mode 2		Mode 3	
Activity	Resource Requiremement (R1)	Duration (day)	Resource Requiremement (R1)	Duration (day)	Resource Requiremement (R1)	Duration (day)
A	4	2	3	3	2	5
B	4	2	2	4	1	6
C	3	2	2	3	1	4
D	3	1	1	3	_	_
E	3	2	2	3	1	4
F	2	1	1	2	_	_
G	2	3	1	5	_	_
H	4	2	3	3	2	4
I	4	3	3	4	2	5
J	4	3	3	4	2	5

The basic DEA code is modified to accept renewable resources by computing the resource consumption of each week similar to the (Weekly constraints) scenario. This means that for any time instance the total resource consumption should not exceed the given threshold level.

For example, let's start with Activity A at day 0. It needs 4 constant resources in mode 1 for 2 days. Now, if Activity B (in mode 1) was started in day 1, it would need 4 additional resources until day 3 (2 days duration). Thus, the resource consumption would be 4, 8 and 4 between days 0-1, 1-2 and 2-3 respectively. If the given resource threshold level was below 8, this scheduling would clearly violate the constraints.

Thus, for every day (time instance) the total resource consumption has to be computed and compared to the threshold level. If AT LEAST one of them violates the threshold limit then penalty is produced. This value is proportional to the maximum of the violations (Appendix G).

$$Penalty = \partial \cdot \max\left(\frac{\text{Maximum of Daily Consumption}}{\text{Resoure Threshold Level}} - 1{,}0\right)$$

First, the network was used to compare DEA with ACO only with threshold R1=6, according to the paper (Zhang, 2012). The results for R1 threshold of 6 are produced by the DEA algorithm as below.

```
DEA optimization started... (Single RENEWABLE Resource with 3 Modes)

Generations:
# 100
…
#1500
Elapsed time is 14.833308 seconds.

Optimal solution:
-----------------
Total Project Time = 12.00
Penalty = 0.00
Fitness = 12.00

Activity    Mode  Time   R1         Critical    Sequence

    A       1      2.0   4.0        yes            1.
    B       2      4.0   2.0        yes            2.
    C       3      4.0   1.0        yes            5.
    D       1      1.0   3.0        no             3.
    E       1      2.0   3.0        yes            4.
    F       1      1.0   2.0        yes            6.
    G       2      5.0   1.0        no             7.
    H       2      3.0   3.0        yes            8.
    I       3      5.0   2.0        yes           10.
    J       1      3.0   4.0        yes            9.

                   Threshold   >=   Consumption
Resource Type  1:  6.00            6.00
```

The evolution of the feasible optimal solution in the function of the generation number (Figure 5.2) and the corresponding Gantt-chart (Figure 5.3) can be seen below. The obtained optimal project time is TPT=12, the same as given in the paper.

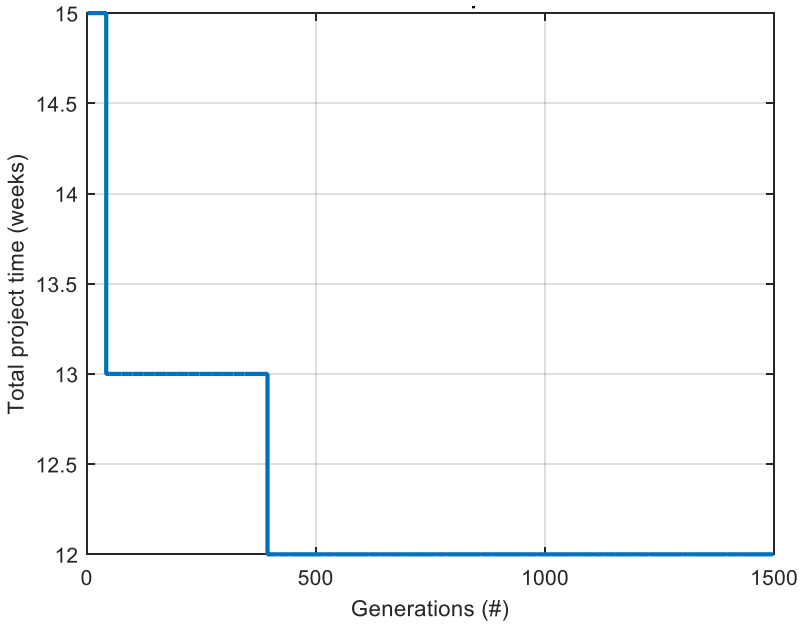


Figure 5.2: Evolution of Feasible Optimal Solution

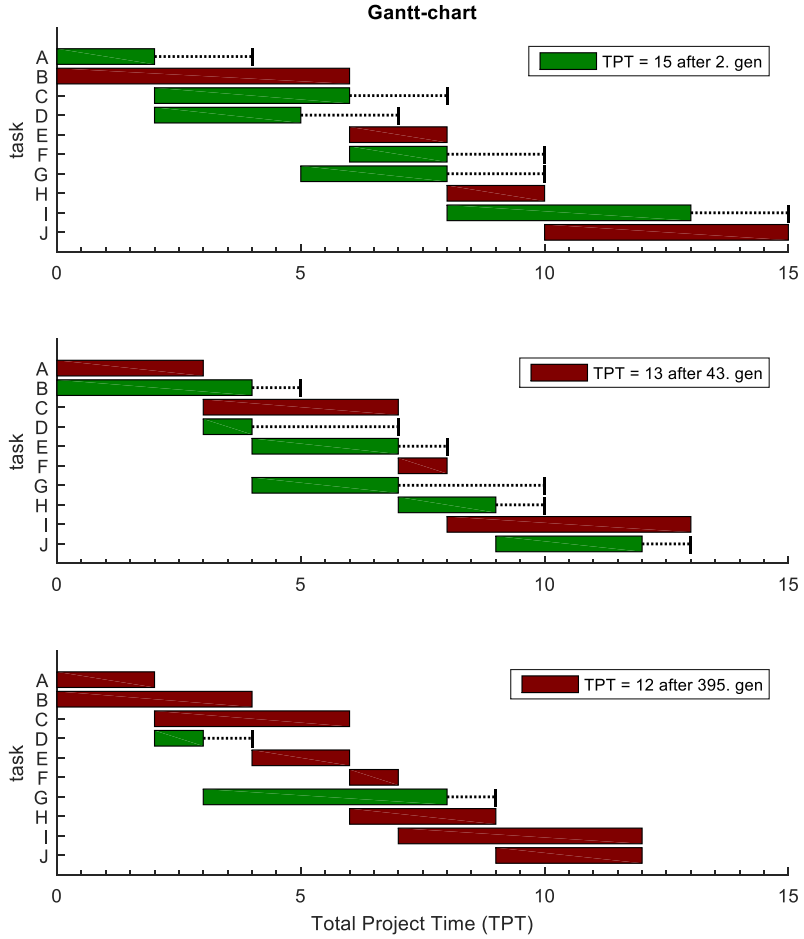


Figure 5.3: Gantt Charts

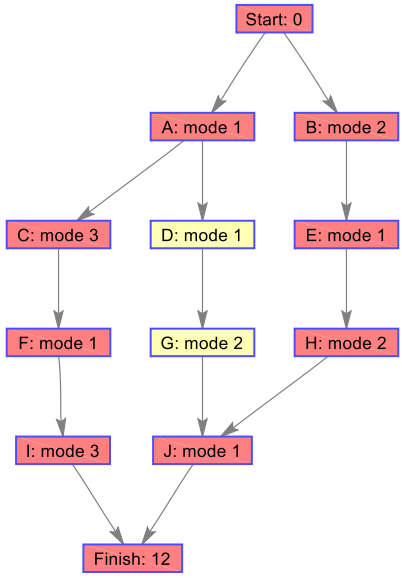


Figure 5.4: Optimal Network Structure

Now, the network is used for comparison by tightening the resource threshold level to R1= 5. This leads to increased difficulty of the problem. In the paper the authors used 100 outer and 20 inner maximum iterations (Genetic Algorithm, Ant Colony Opt., Particle Swarm Opt.) which corresponds to 20*100 = 2000 iterations of the DEA algorithm. The population number was 10 and a total of 40 runs was made. All the other parameters cannot be directly matched with the parameters of the DEA algorithm. The code producing the results is presented in (Appendix H).

Output:

```
OPT   1/ 40: tpt =  15, succ = 1, et = 13.1 s
OPT   2/ 40: tpt =  15, succ = 1, et = 13.1 s
OPT   3/ 40: tpt =  15, succ = 1, et = 13.4 s
OPT   4/ 40: tpt =  15, succ = 1, et = 13.0 s
OPT   5/ 40: tpt =  15, succ = 1, et = 13.5 s
OPT   6/ 40: tpt =  15, succ = 1, et = 13.3 s
OPT   7/ 40: tpt =  14, succ = 1, et = 13.3 s
OPT   8/ 40: tpt = NaN, succ = 0, et = 13.2 s
OPT   9/ 40: tpt =  15, succ = 1, et = 13.6 s
OPT  10/ 40: tpt =  15, succ = 1, et = 13.3 s
OPT  11/ 40: tpt =  15, succ = 1, et = 13.1 s
OPT  12/ 40: tpt =  14, succ = 1, et = 12.5 s
OPT  13/ 40: tpt =  15, succ = 1, et = 13.6 s
```

```
OPT  14/ 40: tpt =  15, succ = 1, et = 13.1 s
OPT  15/ 40: tpt =  15, succ = 1, et = 13.1 s
OPT  16/ 40: tpt =  14, succ = 1, et = 13.3 s
OPT  17/ 40: tpt =  15, succ = 1, et = 13.1 s
OPT  18/ 40: tpt = NaN, succ = 0, et = 13.1 s
OPT  19/ 40: tpt =  15, succ = 1, et = 12.5 s
OPT  20/ 40: tpt =  14, succ = 1, et = 13.1 s
OPT  21/ 40: tpt =  15, succ = 1, et = 13.1 s
OPT  22/ 40: tpt =  15, succ = 1, et = 13.1 s
OPT  23/ 40: tpt =  15, succ = 1, et = 13.1 s
OPT  24/ 40: tpt =  15, succ = 1, et = 12.5 s
OPT  25/ 40: tpt =  15, succ = 1, et = 13.2 s
OPT  26/ 40: tpt =  15, succ = 1, et = 13.2 s
OPT  27/ 40: tpt =  15, succ = 1, et = 13.3 s
OPT  28/ 40: tpt =  17, succ = 1, et = 13.5 s
OPT  29/ 40: tpt =  14, succ = 1, et = 13.1 s
OPT  30/ 40: tpt =  15, succ = 1, et = 13.2 s
OPT  31/ 40: tpt =  15, succ = 1, et = 13.1 s
OPT  32/ 40: tpt =  15, succ = 1, et = 12.4 s
OPT  33/ 40: tpt =  15, succ = 1, et = 13.1 s
OPT  34/ 40: tpt = NaN, succ = 0, et = 13.2 s
OPT  35/ 40: tpt =  15, succ = 1, et = 13.1 s
OPT  36/ 40: tpt =  14, succ = 1, et = 13.1 s
OPT  37/ 40: tpt =  15, succ = 1, et = 13.2 s
OPT  38/ 40: tpt =  15, succ = 1, et = 13.1 s
OPT  39/ 40: tpt =  15, succ = 1, et = 13.0 s
OPT  40/ 40: tpt =  15, succ = 1, et = 13.0 s
```

At some runs DEA was not able to find any (local) optimal solution (in orange). It found the global optimum at runs indicated in red. At the blue runs the algorithm was stuck at a relatively high makespan; however, all the other solutions showed 15 days.

Table 5.2 shows the results. Minimum and Average TPT is the minimum and average makespan achieved in the 40 runs. Success rate is if the algorithm produces a feasible (local) optimum solution but not necessarily the global solution. Computation Time is the average time needed for 1 run.

Table 5.2: Benchmark Results

Methods	Minimum TPT	Average TPT	Success Rate	Comp. Time
GA	14	15.4	75 %	18.3
PSO	14	14.7	75 %	17.2
ACO	14	14.4	81 %	18.0
DEA	14	14.89	92.50 %	13.11

It can be seen that DEA is better than GA and worse than ACO and PSO. Note that the fine tuning of the DEA parameters (such as setting $\partial, C_r, A, \#\ of\ generations..$) might improve the results. DEA has almost always found a feasible solution by comparing to the other methods and found the global optimum six times (6/40 = 15% of the runs).

This low value can be explained by the difficulty of the problem, which DEA seems to be unable to handle. The computational time cannot be compared to the other methods because the simulations were performed on a different (and latest) computer.

5.2 Principle of Pareto Frontier

Pareto frontier is the result of a multi objective optimum design. It is the limit of the feasible design space (Figure 5.5). Any point on this frontier is such that there is no other feasible solution that reduces at least one objective function without increasing another one. On the graph, point A is considered to be the optimal solution because it is located on the pareto frontier. A1 is a better solution than A but unfeasible. A2 is a worse solution than A but feasible.

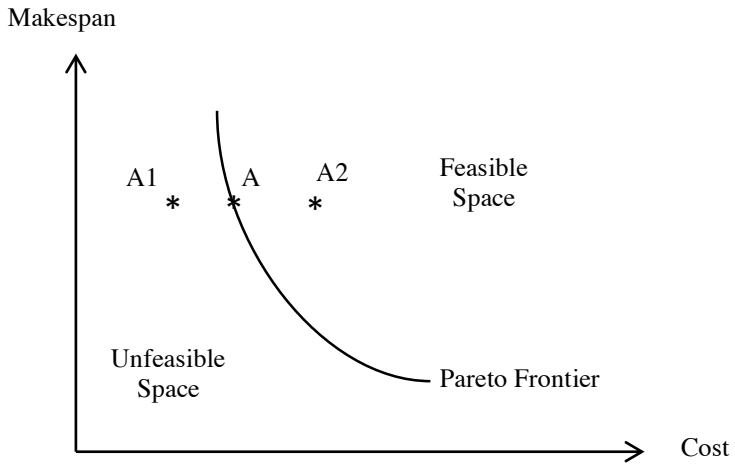


Figure 5.5: Pareto Fronier Feasible Space

5.3 Pareto Frontier for MMSNR Scheduling Problem

The basic definition of the Pareto frontier is that it consists of exactly those alternatives that are *not* dominated by any other alternative. We say that an alternative A dominates B if A outscores B regardless of the tradeoff between time and cost that is, if A is both better and cheaper than B.

A simple algorithm to find the other alternatives on the Pareto frontier is to first sort the alternatives according to one of the objectives, cost. One then starts with the cheapest alternative (which, as noted, always belongs in the Pareto frontier) and skips successive

alternatives in order of increasing cost until one finds one with a higher value. This alternative is then added to the frontier and the search is restarted from it.

A step-by-step description of the algorithm, assuming that *A1,...,An* are the alternatives in increasing order of cost, goes like this:

1. Let *A1*,…, *An* be the alternatives sorted in order of increasing cost/time ratio. Let *i*:=1. Let *P*:={∅}, where ∅ denotes the combination containing no alternatives.
2. For each combination $C \in P$, let $C* := C \cup \{Ai\}$. If $C*$ is not dominated by any combination already in *P*, add $C*$ to *P*.
3. If *i*=n, stop. Otherwise increment *i* by one and repeat from step 2.

In algorithm *B*, we don't need to compare $C*$ to every combination in *P*; it's enough to check whether $C*$ is dominated by the most expensive combination in *P* that is cheaper than $C*$ (Karonen, 2012).

In the following Figures 5.6 to 5.15, around 500 feasible solutions (in blue) (i.e., solutions which satisfy precedence constraints) according to the 14 tasks network presented in Figure 4.1 were plotted according to the pareto frontier algorithm. The solutions (in red) are the best solutions. The black curve is the pareto frontier that connects the best solutions. Here are the results for 10 runs:

Run #1:

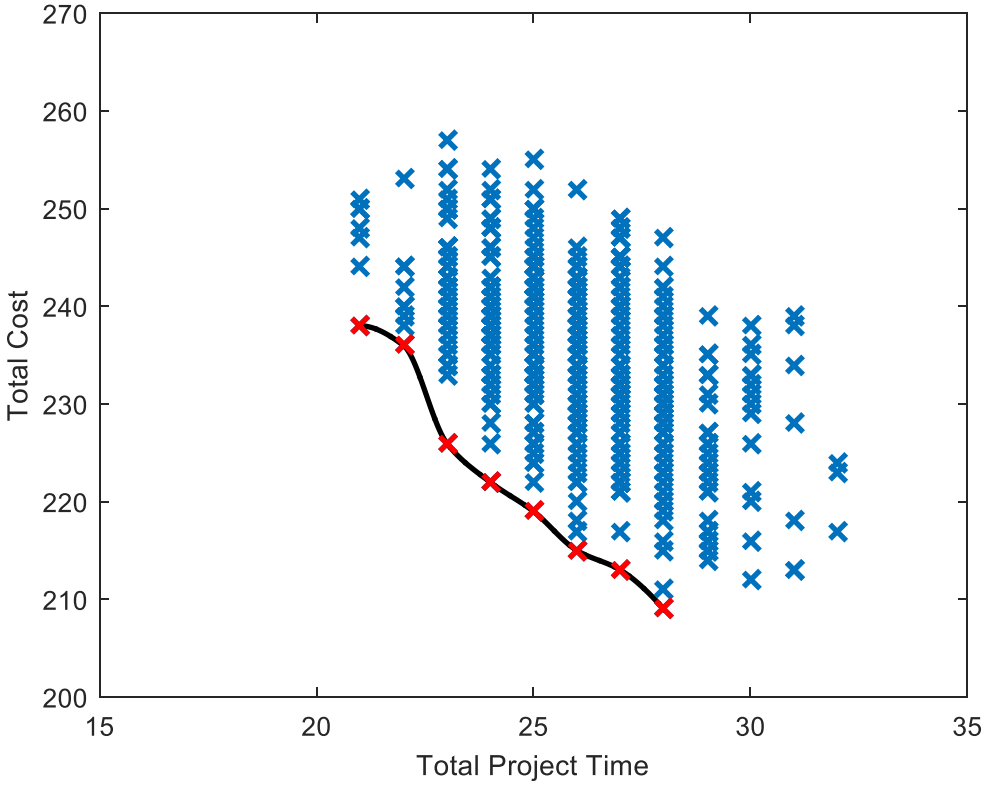


Figure 5.6: Pareto Frontier (Run #1)

Pareto Set:

TPT	Cost
21	238
22	236
23	226
24	222
25	219
26	215
27	213
28	209

Population number: 490

Run #2:

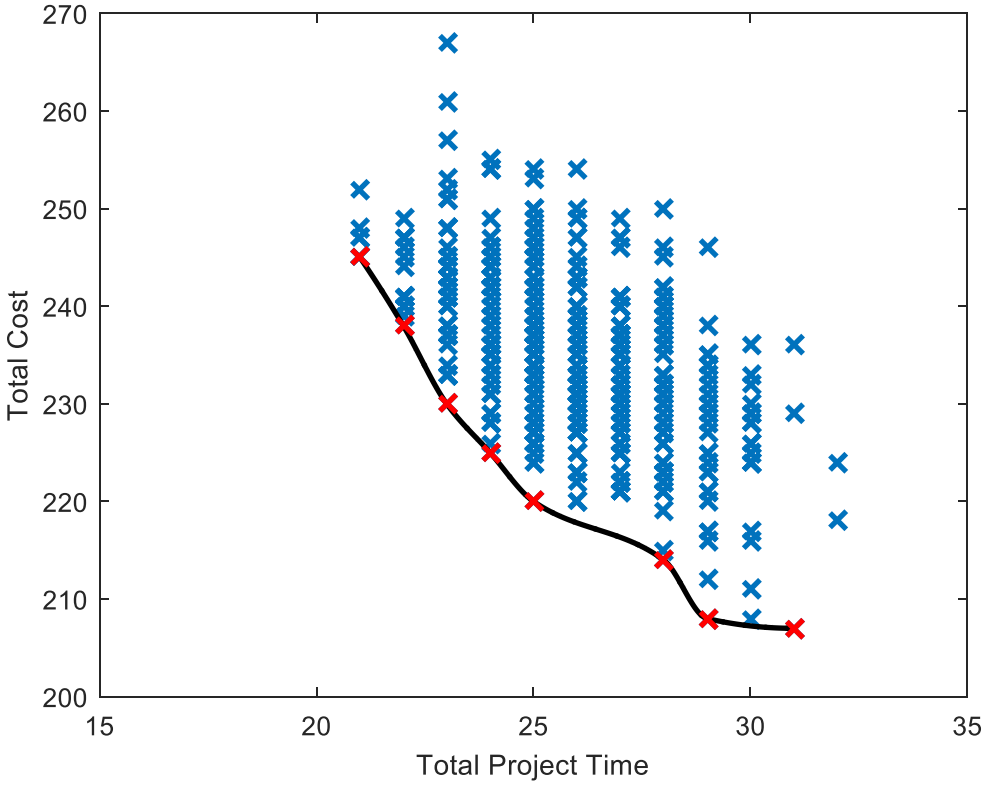


Figure 5.7: Pareto Frontier (Run #2)

Pareto Set:

TPT	Cost
21	245
22	238
23	230
24	225
25	220
28	214
29	208
31	207

Population number: 494

Run #3:

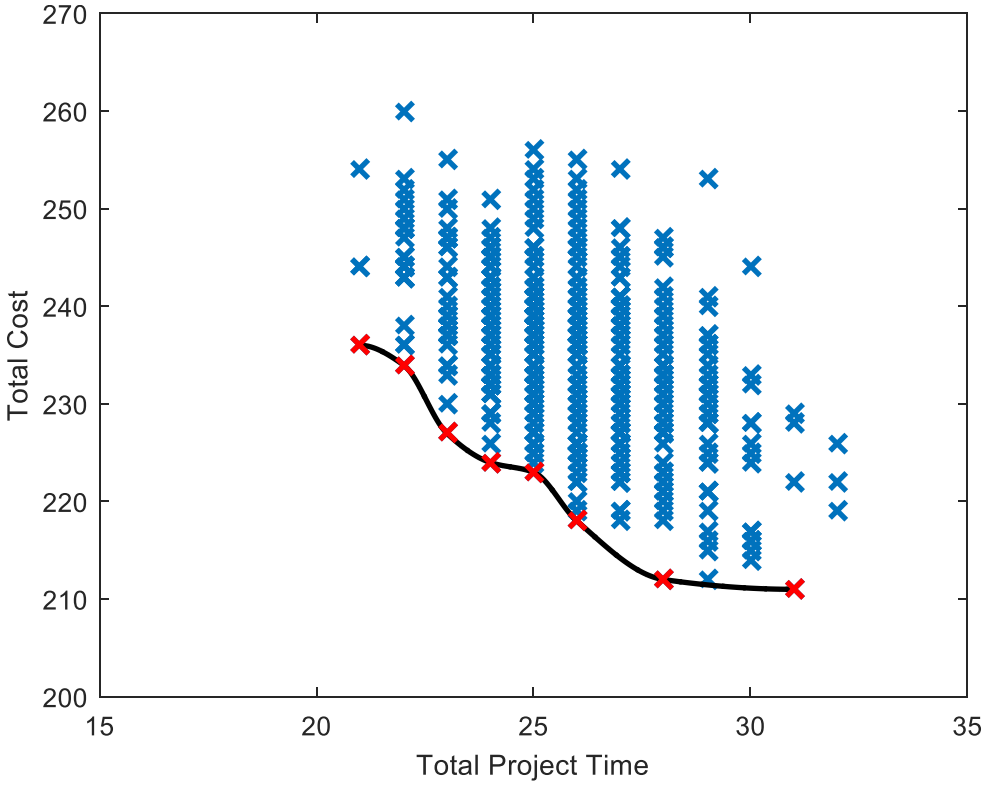


Figure 5.8: Pareto Frontier (Run #3)

Pareto Set:

TPT	Cost
21	236
22	234
23	227
24	224
25	223
26	218
28	212
31	211

Population number: 490

Run #4:

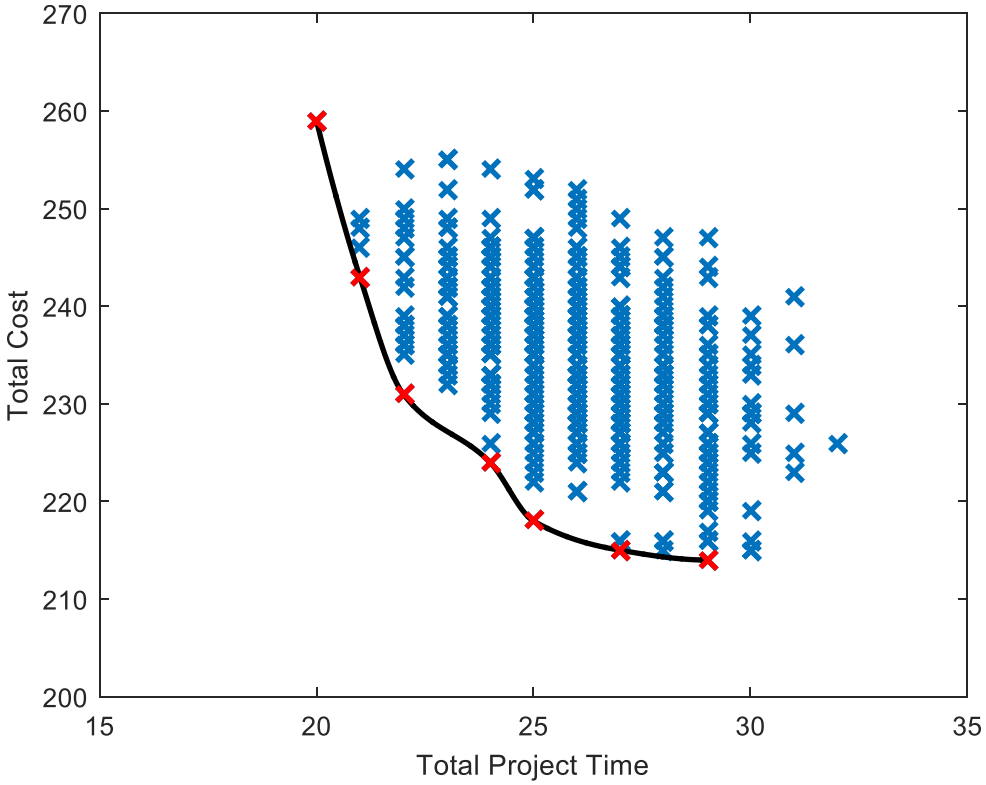


Figure 5.9: Pareto Frontier (Run #4)

Pareto Set:

TPT	Cost
20	259
21	243
22	231
24	224
25	218
27	215
29	214

Population number: 492

Run #5:

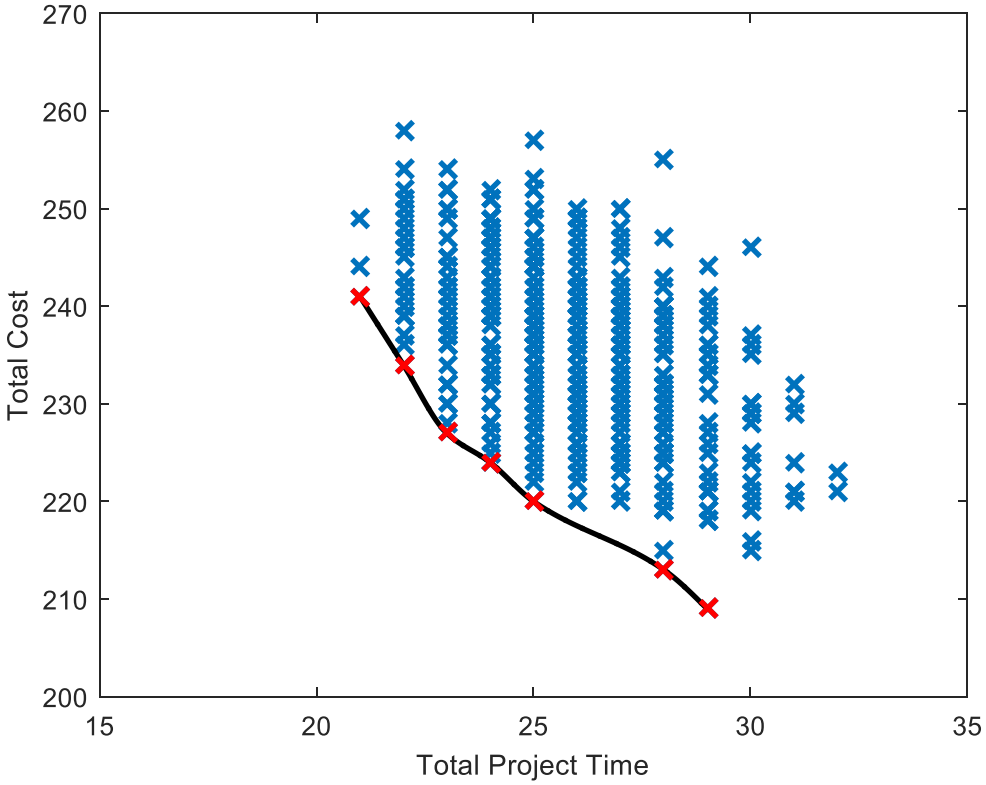


Figure 5.10: Pareto Frontier (Run #5)

Pareto Set:

TPT	Cost
21	241
22	234
23	227
24	224
25	220
28	213
29	209

Population number: 487

Run #6:

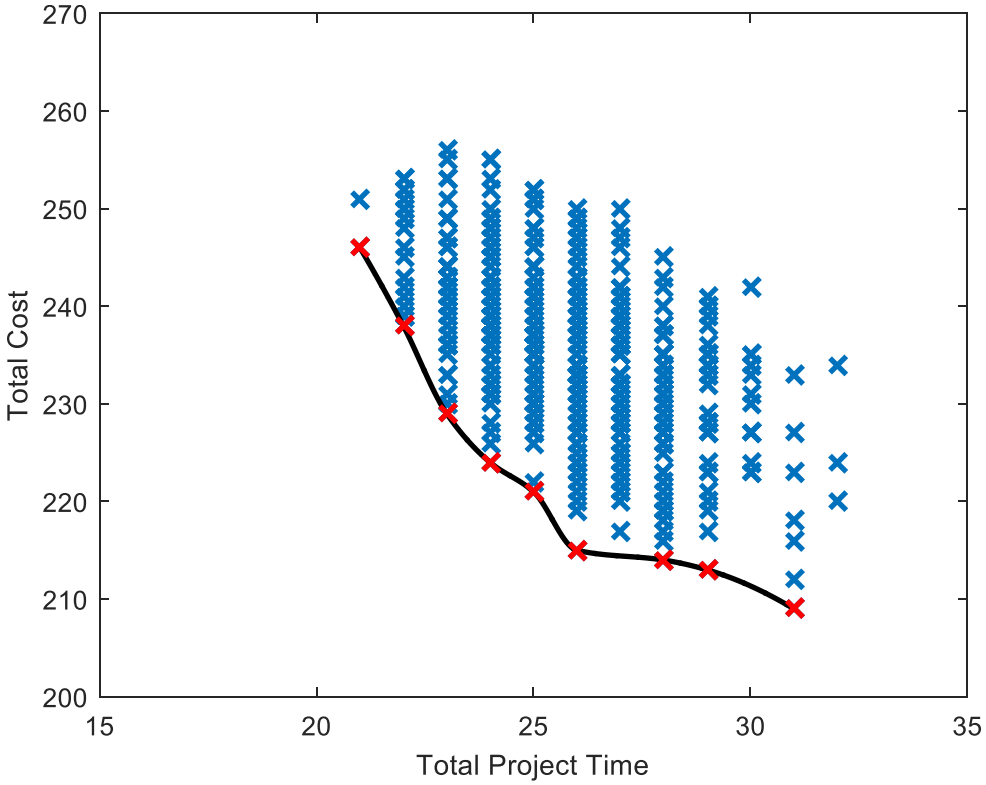


Figure 5.11: Pareto Frontier (Run #6)

Pareto Set:

TPT	Cost
21	246
22	238
23	229
24	224
25	221
26	215
28	214
29	213
31	209

Population number: 494

Run #7:

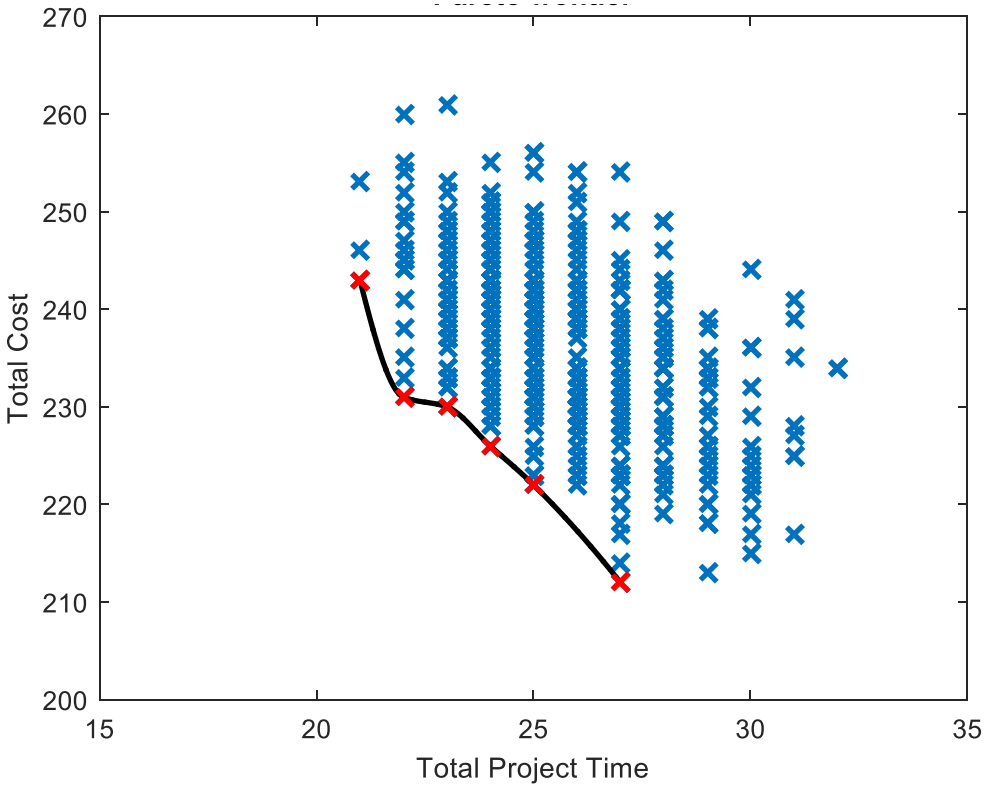


Figure 5.12: Pareto Frontier (Run #7)

Pareto Set:

TPT	Cost
21	243
22	231
23	230
24	226
25	222
27	212

Population number: 493

Run #8:

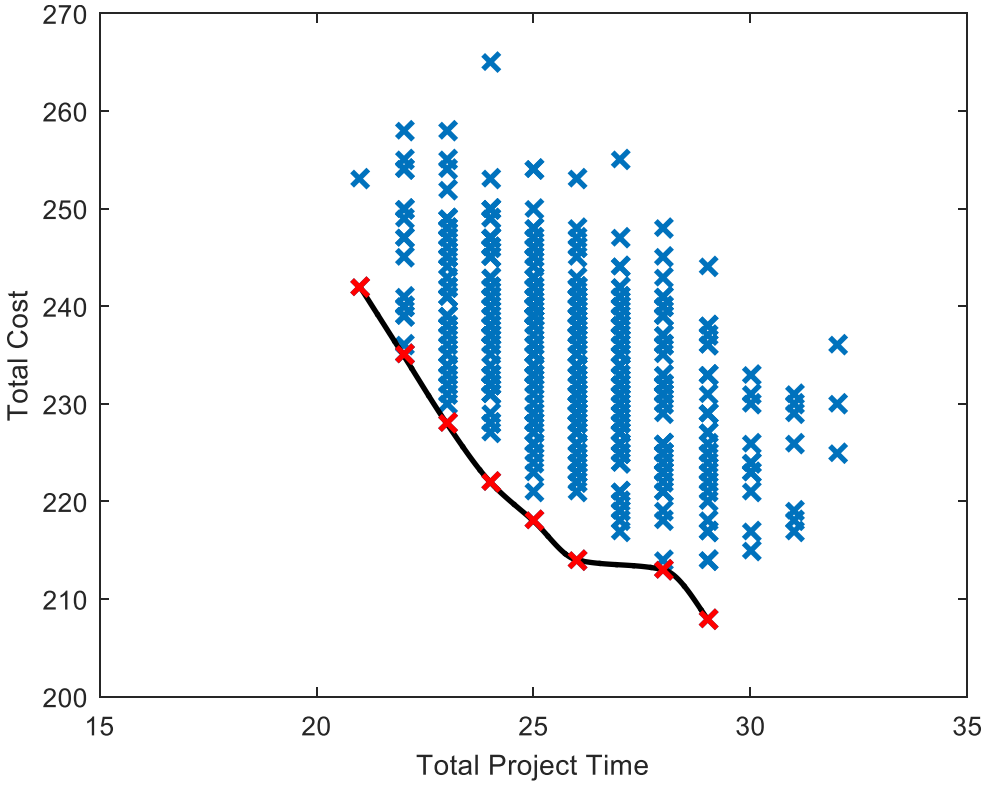


Figure 5.13: Pareto Frontier (Run #8)

Pareto Set:

TPT	Cost
21	242
22	235
23	228
24	222
25	218
26	214
28	213
29	208

Population number: 491

Run #9:

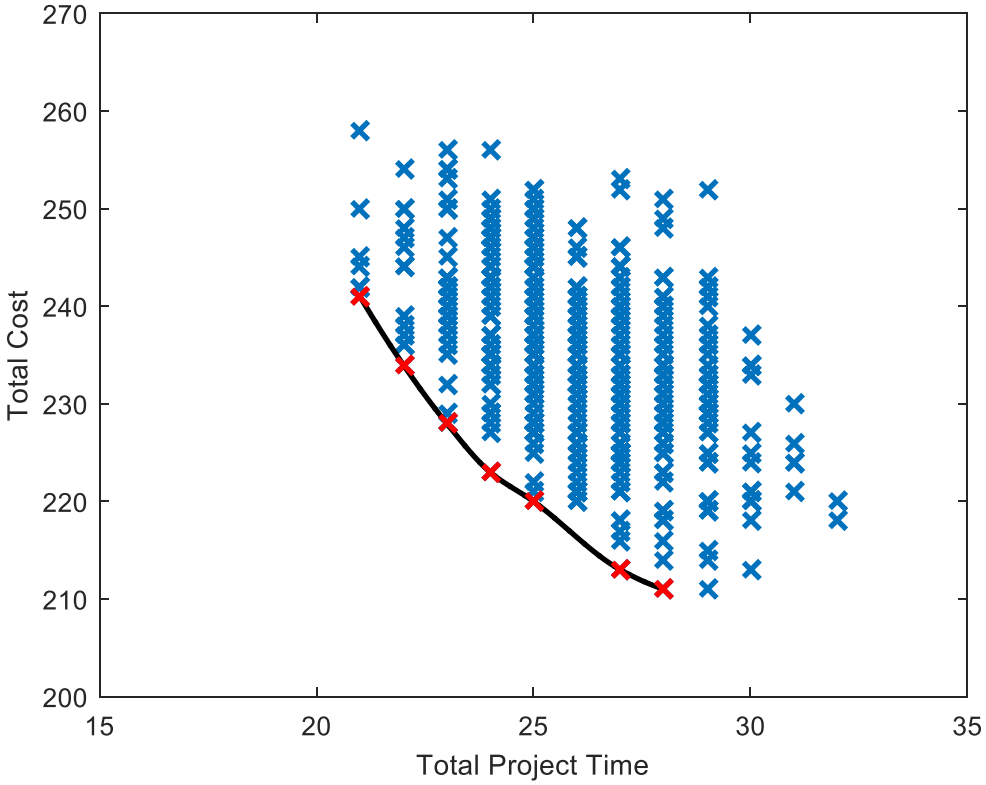


Figure 5.14: Pareto Frontier (Run #9)

Pareto Set:

TPT	Cost
21	241
22	234
23	228
24	223
25	220
27	213
28	211

Population number: 492

Run #10:

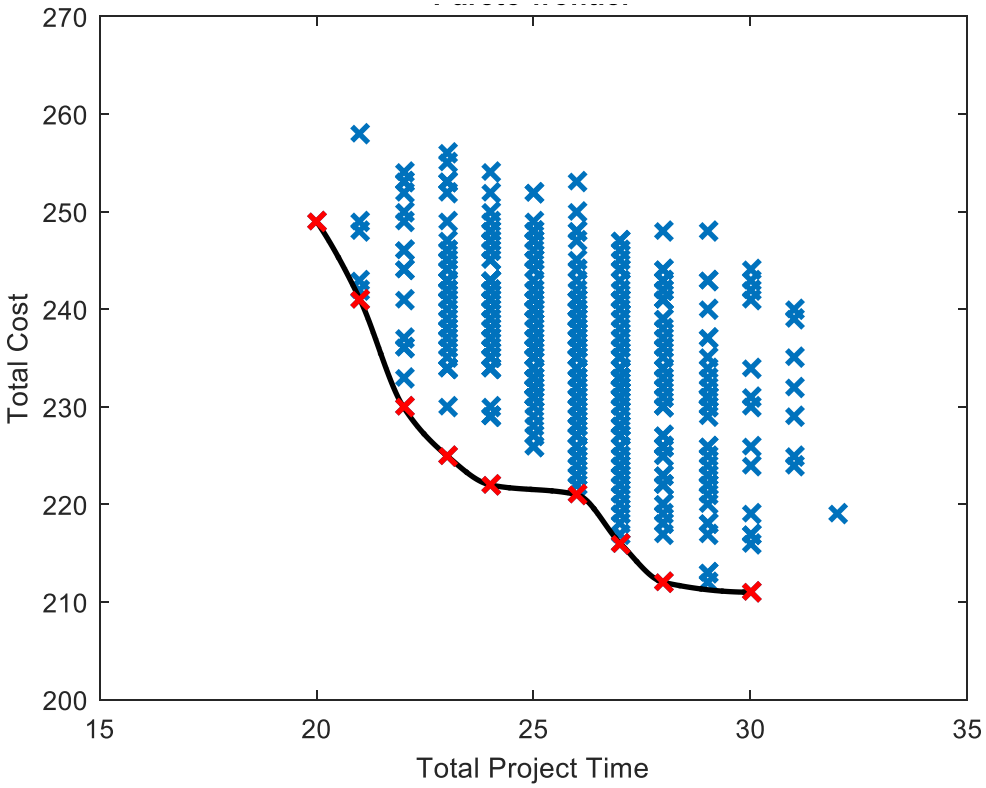


Figure 5.15: Pareto Frontier (Run #10)

Pareto Set:

TPT	Cost
20	249
21	241
22	230
23	225
24	222
26	221
27	216
28	212
30	211

Population number: 492

CHAPTER 6

CONCLUSION

Finding an optimal schedule is often confounded not only by meeting existing constraints but also by adapting to additional constraints and changes to the problem structure. This dissertation attempts to provide optimum solutions to solving complicated project scheduling problems due to resource constraints. The major methodology used in this dissertation is the DEA, which mimics biological phenomena more than mathematical formulation. So far, satisfactory results have been achieved. Future research is contemplated to tackle more difficult situations such as probabilistic resource data, dual thresholds for project cost and time and mulile budget drops.

Differential evolution was used to solve the project scheduling problem for a proposed network that consists of 14 tasks in three different modes. Differential evolution consists of four steps: population structure, mutation, crossover and selection (Figure 6.1). Previous chapters explained the steps and showed how the best vector was selected for the follow-up generation.

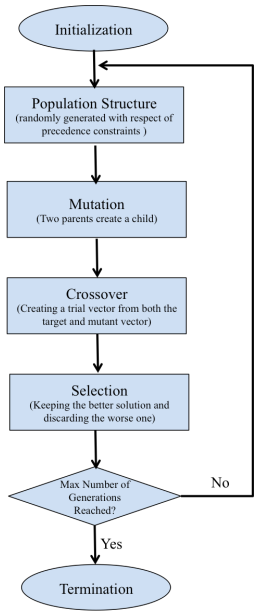


Figure 6.1: DEA Flowchart

The Multi Mode Single Non Renewable (MMSNR) Resource Constrained Project Scheduling Problem involves one resource (cost) in three modes. By using the differential evolution algorithm, the optimal solution indicated the best mode for each activity to be executed and determined the optimal total cost and total project time. The objective was to find the minimium makespan without exceeding the limit of mode 2 duration which is 25 weeks (without optimization) while taking into consideration the precedense rule. The optimal solution was determined to be 20 weeks to complete the project.

In the Multi Mode Multiple Non Renewable (MMMNR) Resource Constrained Project Scheduling Problem the project scheduling problem involves three resources (cost, work hours, material quantity) in three modes. Without optimization, the total duration would have been 25 (using normal execution). Using DEA, the optimal solution was determined to be 24 weeks.

Each of the MMSNR and MMMNR project scheduling problems experienced a weekly constraint. The optimal solution was found by applying DEA. The weekly constraint resulted in more total project time. Then each problem MMSNR and MMMNR faced a cost and time contingency. The scenario assumed a budget drop during week 9, resulting in applying DEA to the remaining tasks to find the optimal solution after the drop incident. In addition, a benchmark problem was presented to compare DEA with Ant Colony Optimization, Genetic Algorithm and Particle Swarm Optimization. DEA reached the same minimum TPT and outperformed in computational time and success rate.

Finally, pareto frontier was investigated, calculating the optimal solutions for a multi objective problem focusing on the tradeoff of the constrained set of parameters.

REFERENCES

AgileCC for AdeptTracker (2008). *AdeptTracker*. Retrieved from http://www.adepttracker.com/agilecc/index.html

Alcaraz, J., Maroto, C., & Ruiz, R. (2003). Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. *Journal of the Operational Research Society*, *54*(6), 614-626.

Bouleimen, K. & Lecocq, H. (2003). A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research*, *149*(2), 268-281.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European journal of operational research*, *112*(1), 3-41.

CC- (M) Pulse. (2016). Soft 112. Retrieved from http://cc-m-pulse.soft112.com

Chen, T., & Zhou, G. (2013). Research on project scheduling problem with resource constraints. *Journal of Software*, *8*(8), 2058-2063.

Concerto Integrated Software Solutions (2016). *Bellrock Group Company*. Retrieved from http://www.concerto.co.uk/about-concerto-integrated-software-solutions/

Damak, N., Jarboui, B., Siarry, P., & Loukil, T. (2009). Differential evolution for solving multi-mode resource-constrained project scheduling problems. *Computers & Operations Research*, *36*(9), 2653-2659.

Ghoddousi, P., Eshtehardian, E., Jooybanpour, S., & Javanmardi, A. (2013). Multi-mode resource-constrained discrete time–cost-resource optimization in project scheduling using non-dominated sorting genetic algorithm. *Automation in construction*, *30*, 216-227.

Hartmann, S. (2001). Project scheduling with multiple modes: a genetic algorithm. *Annals of*

Operations Research, *102*(1-4), 111-135.

Holland, J. H. (1975). Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence. *Ann Arbor, MI: University of Michigan Press*.

Jarboui, B., Damak, N., Siarry, P., & Rebai, A. (2008). A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Applied Mathematics and Computation*, *195*(1), 299-308.

Karonen, I. (2012). How to Compute the Pareto Frontier, Intuitively Speaking?. *Mathematics*. Retrieved from http://math.stackexchange.com/questions/101125/how-to-compute-the-pareto-frontier-intuitively-speaking

Kelley Jr, J. E., & Walker, M. R. (1959). Critical-path planning and scheduling. In *Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference* (pp. 160-173). ACM.

Lancaster, J., & Ozbayrak, M. (2007). Evolutionary algorithms applied to project scheduling problems—a survey of the state-of-the-art. *International journal of production research*, *45*(2), 425-450.

Lewis, J. (2005). *Project Planning, Scheduling & Control, 4E*. McGraw-Hill Pub. Co.

Lewis, J. (2010). *Project Planning, Scheduling, and Control: The Ultimate Hands-On Guide to Bringing Projects in On Time and On Budget: The Ultimate Hands-On Guide to Bringing Projects in On Time and On Budget*. McGraw Hill Professional.

Mori, M., & Tseng, C. C. (1997). A genetic algorithm for multi-mode resource constrained project scheduling problem. *European Journal of Operational Research*, *100*(1), 134-141.

Oracle's Primavera P6 Professional Project Management. (2016). *Oracle*. Retrieved from https://www.oracle.com/applications/primavera/products/project-management.html

Payne, T., Roets, C., & Schlanderer, D. (2014). SAP. *TechTarget*. Retrieved from http://searchsap.techtarget.com/definition/SAP

PD-Trak. (2016). *Project Portfolio Management*. Retrieved from https://pd-trak.com/about-pd-trak/

ProChain Solutions Inc. (2016). Retrived from https://www.prochain.com/about/about.html

Project Management Software. (2016). *Agile Factory*. Retrieved from http://www.agile-factory.com/software/ProjectManagement/projectmanagement.htm

Siemens PLM Software. (2016). *Siemens*. Retrieved from https://www.plm.automation.siemens.com/en_us/plm/

Sprecher, A. (2012). *Resource-constrained project scheduling: Exact methods for the multi-mode case* (Vol. 409). Springer Science & Business Media.

Sprecher, A., & Drexl, A. (1998). Solving multi-mode resource-constrained project scheduling problems by a simple, general and powerful sequencing algorithm. *European Journal of Operational Research*, 431-50.

Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, *11*(4), 341-359.

Van Peteghem, V., & Vanhoucke, M. (2010). A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, *201*(2), 409-418.

Weglarz, J. (Ed.). (2012). *Project scheduling: recent models, algorithms and applications* (Vol. 14). Springer Science & Business Media.

Zhou, J., Love, P. E., Wang, X., Teo, K. L., & Irani, Z. (2013). A review of methods and algorithms for optimizing construction scheduling. *Journal of the Operational Research Society*, *64*(8), 1091-1105.

Zhang, H. (2011). Ant colony optimization for multimode resource-constrained project scheduling. *Journal of Management in Engineering*, *28*(2), 150-159.

Appendix A

Low-level main engine:

```
%% 1) Forward propagation: topological sort + distances
preced = zeros(1,N);
indeg  = full(sum(A));  % A is full adjacency matrix with N nodes
while 1,
    ix = find(indeg == 0);
    if ix == N,
        break;
    else
        for i = 1:length(ix),
            nbor = find(A(ix(i),:) == 1);
            for j = 1:length(nbor),
                if dist(ix(i)) + actt(nbor(j)) >  dist(nbor(j)),
                    preced(1,nbor(j)) = ix(i);
                end
                dist(nbor(j)) = max(dist(nbor(j)),dist(ix(i)) + actt(nbor(j)));
                indeg(nbor(j)) = indeg(nbor(j)) - 1;
            end
            indeg(ix(i)) = indeg(ix(i)) - 1;
        end
    end
end

%% 2) Longest paths in DAG from source to all node
longestPath = zeros(N+1,N);
longestPath(1,1:N) = 1:N;
for i = N:-1:1,
    k = 1;
    currNode = i;
    while 1,
        k = k + 1;
        prevNode  = preced(currNode);
        if prevNode == 0,
            break
        else
            longestPath(k,i) = prevNode;
            currNode         = prevNode;
        end
    end
end

% Total project time
tpt = dist(N);

%% 3) Back propagation for slack time calculation
At = A';
late = tpt*ones(size(actt));
indeg = full(sum(At));
while 1,
    ix = find(indeg == 0);
    if ix == 1,
        break
    else
        for i = 1:length(ix),
            nbor = find(At(ix(i),:) == 1);
            for j = 1:length(nbor),
                late(nbor(j))  = min(late(nbor(j)),late(ix(i)) - actt(nbor(j)));
                indeg(nbor(j)) = indeg(nbor(j)) - 1;
            end
            indeg(ix(i)) = indeg(ix(i)) - 1;
        end
    end
end
slack = late - dist + actt;
```

High-level main engine:

```
for i = 1:Ng, % for each generation

    % 1) Get parents and target vector
    % Random selection (Damak's)
    rix = randperm(Ni);
    rix = rix(1:4);
    PT  = pop(rix);
    C0  = PT{1}; C1 = PT{2}; C2 = PT{3}; T = PT{4}; Tix = rix(4);

    % 2) Mutation
    M = mutation(C0,C1,C2,A);

    % 3) Crossover (get solution vector)
    S = crossover(M,T,G,Cr,Cm,Nm);

    % 4) Fitness evaluation
    fC = fitness(T,G,TimeTable,ResourceTable,Nr,delta,Nm);
    fS = fitness(S,G,TimeTable,ResourceTable,Nr,delta,Nm);

    % 5) Selection
    if fS <= fC,
        pop{Tix} = S;
    end

end
```

Appendix B

```
% Get current resource consumptions FOR ALL resource types
% Nm = number of modes
% Nr = number of resources
for i = 1:length(Nr),
    for j = 1:length(modeVec),
        idxs = (i-1)*Nm + 1;
        idxf = (i-1)*Nm + Nm;
        rtTruncated = ResourceTable(idxs:idxf,:);
        R(i,j) = rtTruncated(modeVec(j),j);
    end
end

% Penalty function (SUM FOR ALL resource types)
Penalty = 0;
for i = 1:length(Nr),
    Penalty = Penalty + delta*max(0,sum(R(i,:))/Nr(i) - 1);
end
```

Appendix C

```
% Penalty function in fitness calculation
Penalty1 = delta*xpenalty(G,T,TimeTable,ResourceTable,wThres,modeVec);
Penalty2 = 0;
for i = 1:length(Nr),
    Penalty2 = Penalty2 + delta*max(0,sum(R(i,:))/Nr(i) - 1);
end
```

```
function p = xpenalty(G,TT,TimeTable,ResourceTable,wThres,modeVec)
%XPENALTY New Penalty Function

WeeklyRes = ResourceTable./TimeTable;
nt = max(max(G)) - 2;
[tpt,~,~,dist,actt] = cpm(G,TT);
GC = [dist-actt dist];
GC = GC(2:end-1,:);
SFmat = zeros(nt,tpt);      % Start 2 Finish matrix (filled with ones)
for i = 1:nt,
    A = WeeklyRes(modeVec(i),i);
    SFmat(i,GC(i,1)+1:GC(i,2)) = A*ones(1,GC(i,2)-GC(i,1));
end
p = sum((sum(SFmat) <= wThres) == 0)/nt;

end
```

Appendix D

```
function p = xpenalty(G,TT,TimeTable,ResourceTable,wThres,modeVec)
%XPENALTY Summary of this function goes here
%   Detailed explanation goes here

WeeklyRes(1:3,:) = ResourceTable(1:3,:)./TimeTable;
WeeklyRes(4:6,:) = ResourceTable(4:6,:)./TimeTable;
WeeklyRes(7:9,:) = ResourceTable(7:9,:)./TimeTable;

nt = max(max(G)) - 2;
[tpt,~,~,dist,actt] = cpm(G,TT);
GC = [dist-actt dist];
GC = GC(2:end-1,:);

% Summary Table 1
SFmat1 = zeros(nt,tpt);     % Start 2 Finish matrix (filled with ones)
for i = 1:nt,
   A = WeeklyRes(modeVec(i),i);
   SFmat1(i,GC(i,1)+1:GC(i,2)) = A*ones(1,GC(i,2)-GC(i,1));
end
p1 = sum((sum(SFmat1) <= wThres(1)) == 0)/nt;

% Summary Table 2
SFmat2 = zeros(nt,tpt);     % Start 2 Finish matrix (filled with ones)
for i = 1:nt,
   A = WeeklyRes(modeVec(i)+3,i);
   SFmat2(i,GC(i,1)+1:GC(i,2)) = A*ones(1,GC(i,2)-GC(i,1));
end
p2 = sum((sum(SFmat2) <= wThres(2)) == 0)/nt;

% Summary Table 3
SFmat3 = zeros(nt,tpt);     % Start 2 Finish matrix (filled with ones)
for i = 1:nt,
   A = WeeklyRes(modeVec(i)+6,i);
   SFmat3(i,GC(i,1)+1:GC(i,2)) = A*ones(1,GC(i,2)-GC(i,1));
end
p3 = sum((sum(SFmat3) <= wThres(3)) == 0)/nt;

% Sum of Penalties
p = p1 + p2 + p3;

end
```

Appendix E

```
function [Gnew,Gnew2,TimeTable,ResourceTable,Rnew,Ropt,gci] = ...
        build2ndNetwork(G,bestInd,Tdrop,Rdrop,TimeTable,ResourceTable,Nr)
%BUILD2NDNETWORK
% Get durations and resource need from the optimal mode vector (1st DEA)
M = bestInd(2,:);
T = size(M);
RR = T;
for i = 1:length(M),
    T(i) = TimeTable(M(i),i);
    RR(i) = ResourceTable(M(i),i);
end
RR = [0 RR 0]';
Ropt = Nr;

% Calculate network with the optimal solution of the first DEA
[~,~,~,dist,actt] = cpm(G,T);

% Find cut-off nodes and the remaining nodes
ix_cut  = find((dist-Tdrop) <= 0);
ix_okay = find((dist-Tdrop) > 0);

% Insert START node (create new network)
Gnew = G(ismember(G(:,1),ix_okay),:);
for i = 1:length(ix_cut),
    cutNode  = ix_cut(i);
    nextNode = G(G(:,1) == cutNode,2);
    for j = 1:length(nextNode),
        if ismember(nextNode(j),ix_okay)
            Gnew(end+1,:) = [1 nextNode(j)];
        end
    end
end

% Re-number network
gci = unique(Gnew);
n = numel(gci);
lut = [(1:n)' gci];
GG = Gnew(:);
for i =1:length(GG),
    GG(i) = lut(lut(:,2) == GG(i),1);
end
Gnew(:) = GG;

% Sort in ascending order
[tmp,ix] = sort(Gnew(:,1));
Gnew = [tmp Gnew(ix,2)];
Gnew2 = Gnew(1:end-1,:)-1;

% Create new time vector
times = actt(ix_okay);
Tnew = [dist-actt dist];
tmp = Tnew(ix_okay,:) - Tdrop;
innx = find(tmp < 0);
times(innx) = times(innx) + tmp(innx,1);
Tnew = times(1:end-1);

% Time Table Update
TimeTable = [zeros(3,1) TimeTable zeros(3,1)];
TimeTable = TimeTable(:,ix_okay);
for i = 1:length(innx),
    TimeTable(:,innx(i)) = times(innx(i))*ones(size(TimeTable,1),1);
end
TimeTable = TimeTable(:,1:end-1);

% Calculate used resources until Tdrop
```

```
R = sum(RR(ix_cut));
RR = RR(ix_okay);
times = actt(ix_okay);
R = R + sum(abs(tmp(innx,1))./times(innx).*RR(innx));

% New resource threshold
Rnew = (Ropt - R) - Rdrop;

% Resource Table Update
ResourceTable = [zeros(3,1) ResourceTable zeros(3,1)];
ResourceTable = ResourceTable(:,ix_okay);
for i = 1:length(innx),
    ccc = (1 - abs(tmp(innx(i),1))./times(innx(i))).*RR(innx(i));
    ResourceTable(:,innx(i)) = ccc*ones(size(ResourceTable,1),1);
end
ResourceTable = ResourceTable(:,1:end-1);

end
```

Appendix F

```
function [Gnew,TimeTable,ResourceTable,Rnew,Ropt,gci] = ...
      build2ndNetwork(G,bestInd,Tdrop,Rdrop,TimeTable,ResourceTable,Nr)
%BUILD2NDNETWORK Build reduced network

Nm = 3; % number of modes

% Get durations and resource need from the optimal mode vector (1st DEA)
M = bestInd(2,:);
T  = NaN(size(M));
RR = NaN(length(Nr),size(M,2));
for i = 1:length(M),
   T(i) = TimeTable(M(i),i);
   RR(i) = ResourceTable(M(i),i);
end
for i = 1:length(Nr),
   for j = 1:length(M),
      idxs = (i-1)*Nm + 1;
      idxf = (i-1)*Nm + Nm;
      rtTruncated = ResourceTable(idxs:idxf,:);
      RR(i,j) = rtTruncated(M(j),j);
   end
end

% Threshold levels
Ropt = Nr;

% Calculate network with the optimal solution of the first DEA
[~,~,~,dist,actt] = cpm(G,T);

% Find cut-off nodes and the remaining nodes
ix_cut  = find((dist-Tdrop) <= 0);
ix_okay = find((dist-Tdrop) > 0);

% Insert START node (create new network)
Gnew = G(ismember(G(:,1),ix_okay),:);
for i = 1:length(ix_cut),
   cutNode  = ix_cut(i);
   nextNode = G(G(:,1) == cutNode,2);
   for j = 1:length(nextNode),
      if ismember(nextNode(j),ix_okay)
         Gnew(end+1,:) = [1 nextNode(j)];
      end
   end
end

% Re-number network
gci = unique(Gnew);
n = numel(gci);
lut = [(1:n)' gci];
GG = Gnew(:);
for i =1:length(GG),
   GG(i) = lut(lut(:,2) == GG(i),1);
end
Gnew(:) = GG;

% Sort in ascending order
[tmp,ix] = sort(Gnew(:,1));
Gnew = [tmp Gnew(ix,2)];
```

```
% Create new time vector
times = actt(ix_okay);
Tnew = [dist-actt dist];
tmp = Tnew(ix_okay,:) - Tdrop;
innx = find(tmp < 0);
times(innx) = times(innx) + tmp(innx,1);
Tnew = times(1:end-1);

% Time Table Update
TimeTable = [zeros(3,1) TimeTable zeros(3,1)];
TimeTable = TimeTable(:,ix_okay);
for i = 1:length(innx),
   TimeTable(:,innx(i)) = times(innx(i))*ones(size(TimeTable,1),1);
end
TimeTable = TimeTable(:,1:end-1);

% Summary Table
GC = [dist-actt dist];
GC = GC(2:end-1,:);
Rused = NaN(size(Nr));
tmpTable = NaN(size(ResourceTable,1),length(ix_okay(1:end-1)-1));
for j = 1:length(Nr),
   Summary = zeros(length(M),max(max(GC)));
   for i = 1:length(M),
      Summary(i,GC(i,1)+1:GC(i,2)) =    ones(1,length(GC(i,1)+1:GC(i,2)))*RR(j,i)/T(i);
   end
   ttt = [zeros(3,1) ResourceTable((j-1)*3+1:(j-1)*3+3,:) zeros(3,1)];
   ttt = ttt(:,ix_okay);
   ix_okay2 = ix_okay-1;
   for i = 1:length(innx),
      icc = ix_okay2(innx(i));
      mult = sum(Summary(icc,Tdrop+1:end) > 0);
      ttt(:,innx(i)) = RR(j,icc)/T(icc)*mult*ones(size(ttt,1),1);
   end
   tmpTable((j-1)*3+1:(j-1)*3+3,:) = ttt(:,1:end-1);
   Rused(1,j) = sum(sum(Summary(:,1:Tdrop)));

end

% New resource table
ResourceTable = tmpTable;

% New resource threshold (Rdrop)
Rnew = ceil((Ropt - Rused)*0.9);
```

Appendix G

```
% Penalty function for RENEWABLE RESOURCE
GC   = [dist-actt+1 dist];
GC   = GC(2:end-1,:);
TCC1 = zeros(length(GC),Cmax);

for i = 1:length(GC),
    TCC1(i,GC(i,1):GC(i,2)) = R(1,i)*ones(1,T(i));
end

RNmax1  = max(sum(TCC1)); % maximum of daily/weekly resource consumption
Penalty = delta*max(0,RNmax1/Nr(1) - 1);
```

Appendix H

```
Ni = 40;     % # of runs
Ng = 2000;   % # of generations
Np = 10;     % population size
Nr = 5;      % threshold of renewable resource

ET = NaN(1,Ni); SC = ET; TP = ET;
for i = 1:Ni,

    [tp,sc,et] = funDEA(Nr,TimeTable,ResourceTable,G,Ng,Np);
    ET(i) = et;
    SC(i) = sc;
    TP(i) = tp;
    fprintf('OPT %3d/%3d: tpt = %3d, succ = %d, et = %2.1f s\n',i,Ni,tp,sc,et);

end
```

VITA

Faisal Manour Altarazi
Department of Mechanical and Aerospace Engineering
238 Kaufman Hall
Norfolk, VA 23529-0247

Education

Old Dominion University, Norfolk, VA, United States — Aug 2012 – May 2017
Degree: Doctor of Philosophy
Major: Mechanical Engineering — Minor: Design and Manufacturing

Gannon University, Erie, PA, United States — Jan 2009 - Dec 2010
Degree: Master of Science
Major: Engineering Management

Riyadh College of Technology, Riyadh, Saudi Arabia — Jan 2001 - Aug 2004
Degree: Bachelor of Engineering Technology
Major: Mechanical Technology — Minor: Production

College of Technology at Jeddah, Jeddah, Saudi Arabia — Nov 1997 - Dec 1999
Degree: An Associate
Major: Mechanical Technology — Minor: Production

Work Experience

Mechanical Design Engineer — Sep 2006 - Apr 2008
Dallah Trans Arabia Co. contract with Saudi Aramco Co., Jeddah, Saudi Arabia

- Worked in the community services department in planning and technical support
- Prepared project packages (drawing, cost estimate, and scope of work)

Process Engineer — Jan - Apr 2006
Al-Tuwairqi Holding Co. in Al-Ittefaq Steel Products Co., Jeddah, Saudi Arabia

- Worked in the production department and the roll shop department
- Worked on the industrial calculations to produce rebar and analyzed problems
- Used AutoCAD to draw mechanical parts and edited the layout of the factory

Training Engineer — Sep - Dec 2005
Jeddah Cable Co., Jeddah, Saudi Arabia

- Supervised and registered new candidates and explained briefly the manufacturing cable process to become machine operators

Conference Presentations

Altarazi, F. & Bao, H. (2015). *Investigating the Impact of Buffer Size in Critical Chain Management*, The International Conference on Flexible Automation and Intelligent Manufacturing (FAIM), University of Wolverhampton, Wolverhampton, UK