

Use of Genetic Algorithms in Multi-Objective Multi-Project Resource  
Constrained Project Scheduling

Fikri Küçüksayacıgil

Submitted to the Graduate School of Engineering and Natural Sciences  
in partial fulfillment of the requirements for the degree of  
Master of Science

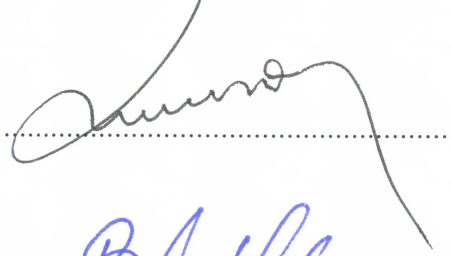
Sabancı University

January, 2014

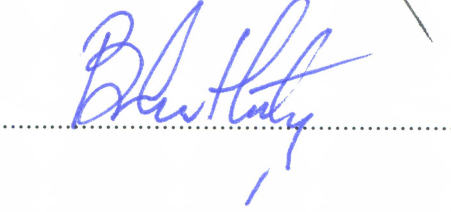
USE OF GENETIC ALGORITHMS IN MULTI-OBJECTIVE MULTI-  
PROJECT RESOURCE CONSTRAINED PROJECT SCHEDULING

APPROVED BY:

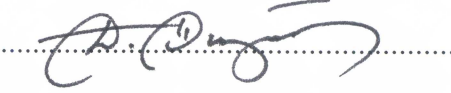
Prof. Dr. Gündüz Ulusoy  
(Thesis Supervisor)



Assoc. Prof. Dr. Bülent Çatay



Asst. Prof. Dr. Demet Özgür Ünlüakın



DATE OF APPROVAL: 07/01/2014

To my wife and family

## **Acknowledgements**

I would like to express my deepest gratitude to my thesis advisors Prof. Dr. Gündüz Ulusoy for his huge and never-ending support. The ingenuity character, the great deal effort, skills, the knowledge and the guidance of Prof. Dr. Gündüz Ulusoy were the factors that kept this research going. Without his hope, motivation, and assistance, this research would have never existed.

Secondly, I am grateful to my beloved wife, Gulnihal, for her early contributions to this research, and her moral support through my thesis. She was always there for me, whenever I needed help.

I am thankful to many of our faculty members for their helpful advices and support. I am also thankful to my dear colleagues, Canan, Gürkan, Murat, Burcu and Ezgi for sharing their experiences and their smile whenever necessary.

Finally, I would like to thank to my family for all the support they have provided. Without their wisdom, I would have never been able to earn a master's degree or write a thesis.

© Fikri Küçüksayacıgil 2014

All Rights Reserved

# Çok Amaçlı Kaynak Kısıtlı Çoklu Proje Çizelgelemede Genetik Algoritmanın Kullanımı

Fikri Küçüksayacıgil

Endüstri Mühendisliği, Yüksek Lisans Tezi, 2012

Tez Danışmanı: Prof. Dr. Gündüz Ulusoy

**Anahtar Kelimeler:** Kaynak Kısıtlı Proje Çizelgeleme Problemi; Genetik Algoritma; Çok Amaçlı Kaynak Kısıtlı Proje Çizelgeleme Problemi; Kaynak Kısıtlı Çoklu Proje Çizelgeleme Problemi; Geriye-ileriye Yöntemi

## Özet

Kaynak kısıtlı proje çizelgeleme problemi, araştırmacılar tarafından, yenilenebilir ve yenilenemez kaynaklar da göz önüne alınarak çokça çalışılmıştır. Çözüm yöntemleri olarak, birçok kesin ve bulgusal yöntem önerilmiştir. İlgili problemin teknik yazında, çok modlu kaynak kısıtlı proje çizelgeleme problemi, çok amaçlı kaynak kısıtlı proje çizelgeleme problem ve kaynak kısıtlı çoklu proje çizelgeleme problem gibi uzantıları çalışılmıştır. Bu çalışmada, çok amaçlı kaynak kısıtlı çoklu proje çizelgeleme problem üzerinde durulmuştur. Çözüm yöntemi olarak, teknik yazında Bastırılmamış Sınıflandırılmalı Genetik Algoritma II (NSGA-II) olarak bilinen algoritma tercih edilmiştir. Çeşitli çaprazlama yöntemleri ve ebeveyn seçim yöntemleri kullanılarak, genetik algoritma parametrelerinin hassas ayarları ayrıntılı bir şekilde yapılmıştır. Bu deneyde, teknik yazında Yanıt Yüzeyi Yöntemi olarak bilinen istatistiksel bir yaklaşım kullanılmıştır. Çözüm kalitesini geliştirmek için, geriye-ileriye (forward-backward) yöntemi hem işlem sonrası aşamada, hem de algoritma devam ederken yeni nüfus üretilmesinde kullanılmıştır. Ek olarak, çeşitli iraksama yöntemleri önerilmiştir ve bunlardan entropi temelli olanı ayrıntılı bir şekilde çalışılmıştır.

Algoritmanın performansı ve çözüm süreleri kaydedilmiştir. Bu çalışmada ayrıca, yeni bir çoklu proje sınamaya dataları üretme yöntemi önerilmiş, sınamaya dataları üretilmiş ve bunlar ile algoritmanın performansı sınanmıştır. Sonuçlar, geriye-ileriye yönteminin çözüm kalitesini artırmada etkili olduğunu göstermiştir.

# Use of Genetic Algorithms in Multi-Objective Multi-Project Resource Constrained Project Scheduling

Fikri Küçüksayacıgil

Industrial Engineering, Master's Thesis, 2014

Thesis Supervisors: Prof. Dr. Gündüz Ulusoy

**Keywords: RCPSP; Genetic Algorithms; Multi-objective RCPSP; Multi-project RCPSP; backward-forward scheduling**

## **Abstract**

Resource Constrained Project Scheduling Problem (RCPSP) has been studied extensively by researchers by considering limited renewable and non-renewable resources. Several exact and heuristic methods have been proposed. Some important extensions of RCPSP such as multi-mode RCPSP, multi-objective RCPSP and multi-project RCPSP have also been focused. In this study, we consider multi-project and multi-objective resource constrained project scheduling problem. As a solution method, non-dominated sorting genetic algorithm is adopted. By experimenting with different crossover and parent selection mechanisms, a detailed fine-tuning process is conducted, in which response surface optimization method is employed. In order to improve the solution quality, backward-forward pass procedure is proposed as both post-processing as well as for new population generation. Additionally, different divergence applications are proposed and one of them, which is based on entropy measure is studied in depth. The performance of the algorithm and CPU times are reported. In addition, a new method for generating multi-project test instances is proposed and the performance of the algorithm is evaluated through test instances generated through this method of data generation. The results show that backward-forward pass procedure is successful to improve the solution quality.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Literature Survey</b>	<b>7</b>
2.1	Single objective . . . . .	7
2.1.1	Exact solution methods . . . . .	7
2.1.1.1	Minimization of Cmax . . . . .	7
2.1.1.2	Maximization of net present value . . . . .	9
2.1.1.3	Other objectives . . . . .	10
2.1.1.4	Other problems . . . . .	10
2.1.2	Heuristic solution methods . . . . .	10
2.1.2.1	Priority rules . . . . .	10
2.1.2.1.1	Minimization of Cmax . . . . .	10
2.1.2.1.2	Maximization of net present value . . . . .	12
2.1.2.2	Forward and backward recursion . . . . .	13
2.1.2.3	Sampling method . . . . .	13
2.1.2.4	Variable neighbourhood search . . . . .	14
2.1.2.5	Other methods . . . . .	14
2.1.2.6	Other objectives . . . . .	15
2.1.3	Metaheuristic solution methods . . . . .	15
2.1.3.1	Genetic algorithm . . . . .	15
2.1.3.1.1	Minimization of Cmax . . . . .	15
2.1.3.1.2	Maximization of net present value . . . . .	18
2.1.3.1.3	Other formulations . . . . .	18
2.1.3.2	Simulated annealing . . . . .	20
2.1.3.3	Tabu search . . . . .	20
2.1.3.4	Other methods called hybrid . . . . .	21
2.1.4	Review papers . . . . .	21

2.2	Multi-objective . . . . .	22
2.2.1	Exact solution methods . . . . .	22
2.2.2	Heuristic methods . . . . .	23
2.2.3	Other formulations . . . . .	29
2.3	Multi-project . . . . .	32
2.3.1	Exact solution methods . . . . .	32
2.3.2	Genetic algorithms . . . . .	33
2.3.3	Hybrid methods . . . . .	34
2.3.4	Priority Rules . . . . .	35
2.3.5	Auction mechanism . . . . .	36
2.3.6	Other methods . . . . .	37
<b>3</b>	<b>A Multi-objective Genetic Algorithm Approach and Extensions</b>	<b>42</b>
3.1	Project instance . . . . .	44
3.1.1	Modifications in single project instances . . . . .	44
3.1.2	Multi-project instance generation . . . . .	46
3.1.2.1	Data generation for testing due date . . . . .	47
3.1.2.2	Data generation for testing lump sum payments . . . . .	51
3.1.2.3	Data generation for testing resource limits . . . . .	52
3.1.2.4	Data generation with different number of activities . . . . .	52
3.2	Preprocessing . . . . .	52
3.3	Forming the initial population . . . . .	53
3.3.1	Precedence feasible activity list . . . . .	54
3.3.2	Mode assignment list . . . . .	54
3.3.2.1	Random mode assignment . . . . .	54
3.3.2.2	The mode with longest duration . . . . .	55
3.3.2.3	The mode with minimum average utilization . . . . .	55
3.3.2.4	Iterative assignment . . . . .	55
3.3.2.5	Local repair . . . . .	56
3.3.2.6	Extensive repair . . . . .	56
3.3.3	Random and feasible initial population . . . . .	56
3.4	Fitness calculation . . . . .	59
3.5	Forming the next generation . . . . .	62
3.5.1	Crossover . . . . .	62
3.5.1.1	One-point crossover . . . . .	62

3.5.1.2	Two-point crossover . . . . .	62
3.5.1.3	Multi component uniform order-based crossover (MCUOBC)	65
3.5.2	Mutation . . . . .	65
3.5.3	Parent selection . . . . .	65
3.5.3.1	Roulette wheel selection . . . . .	67
3.5.3.2	Binary tournament selection . . . . .	69
3.5.4	Population reduction . . . . .	69
3.5.4.1	The best ones . . . . .	70
3.5.4.2	Random reduction . . . . .	70
3.5.4.3	Elitist reduction . . . . .	70
3.6	Non-dominated sorting procedure . . . . .	70
3.7	Crowding distance . . . . .	71
3.8	Basic scheme . . . . .	73
<b>4</b>	<b>Fine-Tuning of the Parameters</b>	<b>75</b>
4.1	Commonly used performance measures in the literature . . . . .	75
4.1.1	The size of the approximation set $M$ . . . . .	75
4.1.2	Distance from the reference set $R$ . . . . .	76
4.1.3	Coverage error . . . . .	76
4.1.4	Error ratio . . . . .	76
4.1.5	Hypervolume . . . . .	77
4.1.6	Binary $\varepsilon$ indicator (Zitzler et al. [155]) . . . . .	77
4.1.7	Maximum spread (Zitzler [152]) . . . . .	77
4.2	Other studies in the literature about performance measurement . . . . .	78
4.3	GA parameter design in the literature . . . . .	79
4.4	Getting the best parameter combination in our study . . . . .	80
4.5	Selecting the best operant combination . . . . .	86
4.6	Fine-tuning process with multi-project instances . . . . .	86
<b>5</b>	<b>Divergence Applications and Local Searches</b>	<b>89</b>
5.1	Divergence applications . . . . .	89
5.1.1	Entropy-based divergence application . . . . .	89
5.1.2	Objective value-based divergence application . . . . .	92
5.1.3	Grid-based divergence application . . . . .	94
5.1.4	Archive-based convergence check . . . . .	95
5.2	Local searches . . . . .	95

5.2.1	Backward and forward pass procedure . . . . .	95
5.2.2	Simulated annealing . . . . .	96
<b>6</b>	<b>Computational Study</b>	<b>102</b>
6.1	Implementation with test instances . . . . .	102
6.1.1	Implementation with the test instances generated in Can and Ulusoy [20] . . . . .	102
6.1.2	Implementation with the test instances generated in this thesis . . . . .	112
6.2	Results . . . . .	121
6.2.1	The test instances generated in Can and Ulusoy [20] . . . . .	122
6.2.2	The test instances generated in this thesis . . . . .	123
6.2.2.1	Bi-objective . . . . .	124
6.2.2.2	Triple-objective . . . . .	125
6.2.2.3	The evaluation of BFP for current objectives . . . . .	126
<b>7</b>	<b>Conclusion</b>	<b>128</b>
	<b>Appendices</b>	<b>148</b>
<b>A</b>	<b>Results of Fine-Tuning Experiments for Single Projects</b>	<b>148</b>
<b>B</b>	<b>Results of Fine-Tuning Experiments for Multi-Project</b>	<b>151</b>

## List of Figures

3.1	Multi-Project Network Structure . . . . .	48
4.1	Hypervolume Indicator for Two Different Cases . . . . .	82
5.1	An Example of an Individual for CFP . . . . .	89
5.2	An Example Calculation of Divergence Measure for RCPSP . . . . .	91
5.3	BFP Procedure . . . . .	97
5.4	General Framework of BFP . . . . .	98
5.5	An Example of Activity Insertion . . . . .	99
6.1	Instance A11_11 . . . . .	105
6.2	Instance A11_21 . . . . .	105
6.3	Instance A21_11 . . . . .	106
6.4	Instance B1010_21 . . . . .	106

## List of Tables

3.1	Merging the Single Projects . . . . .	47
3.2	The Earliest Finishing Times of The Projects . . . . .	48
3.3	Average Usage of Renewable Resource . . . . .	49
3.4	Average Usage of Nonrenewable Resource . . . . .	49
3.5	The Npv of The Projects . . . . .	50
4.1	Design of Experiment . . . . .	81
4.2	The Result of the Experiment 1 . . . . .	85
4.3	The Best Parameter Combinations . . . . .	88
6.1	Threshold and Stoppage Values for BFP Procedure . . . . .	104
6.2	Abbreviations Used in Tables . . . . .	105
6.3	Results for Set A - BFP in the Final Stage . . . . .	105
6.4	Results for Set B - BFP in the Final Stage . . . . .	106
6.5	Results for Set C - BFP in the Final Stage . . . . .	106
6.6	Paired t-test Result for Set A - BFP in the Final Stage . . . . .	107
6.7	Paired t-test Result for Set B - BFP in the Final Stage . . . . .	107
6.8	Paired t-test Result for Set C - BFP in the Final Stage . . . . .	108
6.9	Results for Set A - BFP in the Intermediate Stages . . . . .	109
6.10	Results for Set B - BFP in the Intermediate Stages . . . . .	109
6.11	Results for Set C - BFP in the Intermediate Stages . . . . .	110
6.12	Frequency of BFP During the Implementation of GA . . . . .	110
6.13	Activity and Mode Entropy Thresholds . . . . .	111
6.14	Results for Entropy-Based Divergence - First Level . . . . .	111
6.15	Results for Entropy-Based Divergence - Second Level . . . . .	111
6.16	Frequency for Entropy-Based Divergence - First Level . . . . .	112
6.17	Frequency for Entropy-Based Divergence - Second Level . . . . .	112
6.18	Results for maxNPV/minMFT . . . . .	114

6.19	CPU Times for maxNPV/minMFT . . . . .	114
6.20	Results for maxNPV/minMWT . . . . .	115
6.21	CPU Times for maxNPV/minMWT . . . . .	115
6.22	Results for maxNPV/minMCT . . . . .	116
6.23	CPU Times for maxNPV/minMCT . . . . .	117
6.24	Results for minCMAX/maxNPV/minRUD . . . . .	117
6.25	CPU Times for minCMAX/maxNPV/minRUD . . . . .	118
6.26	Results for minCMAX/maxNPV/maxMO . . . . .	118
6.27	CPU Times for minCMAX/maxNPV/maxMO . . . . .	118
6.28	Threshold and Stoppage Values for Different Objective Combinations . . . . .	119
6.29	The Result of BFP Procedure for maxNPV/minMFT . . . . .	120
6.30	The Result of BFP Procedure for maxNPV/minMWT . . . . .	120
6.31	The Result of BFP Procedure for maxNPV/minMCT . . . . .	120
6.32	The Result of BFP Procedure for minCMAX/maxNPV/minRUD . . . . .	121
6.33	The Result of BFP Procedure for minCMAX/maxNPV/maxMO . . . . .	121
A.1	Result of the Experiment for Single Projects - 1 . . . . .	149
A.2	Result of the Experiment for Single Projects - 2 . . . . .	150
A.3	Result of the Experiment for Single Projects - 3 . . . . .	150
B.1	Result of the Experiment for Multiple Projects - 1 . . . . .	152
B.2	Result of the Experiment for Multiple Projects - 2 . . . . .	153
B.3	Result of the Experiment for Multiple Projects - 3 . . . . .	154

# Chapter 1

## Introduction

With the advent of humankind and increasing economic activity in global scale we observe projects increasingly more for accomplishing large complex work. For example, large infrastructure construction such as building subways, dams, and suspension bridges and complex, knowledge based work such as research and development (R&D), new product development (NPD) and software development are all organised as projects. Finishing these projects on time, within budget and meeting required quality requirements is a major task indeed providing a great challenge for the project owners as well as the project managers alike.

In the last decades an extensive amount of work has been accomplished in developing exact and heuristic algorithms for the solution of resource constrained project scheduling problem (RCPSP) for single as well as multiple projects. In recent years, there has been considerable improvement in the development of methods emulating the real life decision environments in project management. Research on multi-objective project scheduling is one such attempt.

Management of multi-project turns out to be always challenging for organizations. However, if the case is to deal with different type of projects for multiple organizations, the complexity increases even more. Managing and tracking different project plans, checking budget and costs, considering different types of resources and materials and communicating with the customers become one of the most challenging tasks of project managers, if the right solutions are not available. Therefore, the right solutions regarding the projects, their schedules and resource profiles should be in hand. For this purpose, many researches have been conducted for obtaining the right solutions, if possible, of project scheduling problems.



Organizations can be divided in two main groups in relation with projects. The first group is called project-driven organizations, whose primary business is made up of projects. This kind of firms measure their growth with the type, size, location and nature of the projects that they conduct. Another measurement of growth is performed with the decisions about how the required resources are provided. The examples of project-driven organization include architect and engineering firms, software development firms and other firms that bid for work on a project-by-project basis. The R&D department of many companies work on project basis as well. The second class of organizations is project-dependent companies. In contrast to project-driven firms, they do not work on project basis. Rather, they provide goods and services as their mainstream business. The projects conducted within project-dependent organizations are generally funded internally. The reason why these companies are grouped as project-dependent is that they depend on projects in order to support their primary lines of business, but the projects are not their principle offering to the marketplace. The examples of these organizations are banking, financial services, governmental agencies, universities, hospitals, etc.

Boctor [11] explains three main organization types for project management. He describes their characteristics, advantages, disadvantages and the environments where they can be seen. The first group is functional type of organization where the project is part of the functional organization of company. A project can be seen in any of the functional part of the company. The second group of organization types is pure project management organization, which is independent from the home organization with its own staff and its own administration. The last organization type is matrix form, which is proposed because of the serious disadvantages of previous forms of organizations. Matrix form organization is developed by combining functional and pure organization types in order to take advantage of their advantages.

RCPSPP can be said to be a generalization of job shop scheduling problem because each activity in a project may use multiple resources and each resource have a capacity larger than one. In this type of problem, each activity requires some amount of limited resources and there are precedence relations between activities. Additionally, preemption of activity is not allowable. The objective which is generally accepted and commonly studied is the minimization of  $C_{max}$ ; that is, total project completion time without violating the resource constraints and precedence relations.

After development of the large scale projects during World War II, many researches have begun to emerge for dealing with project scheduling. Initially, the main purpose of the studies was to determine the start times of the activities and to obey the precedence

relations such that  $C_{max}$  can be minimized. This effort has led to Critical Path Method (CPM) and Program Evaluation and Review Technique (PERT). While CPM deals with project which has deterministic task duration, PERT requires the project to have stochastic task durations and tries to determine probabilistic manner of the completion time of the project.

Whereas there is an unconstrained version of above mentioned problem, resource constraints have been currently taken into consideration because it can be thought as more realistic. Additionally, each activity may have multiple execution modes. A mode is an alternative way of executing of an activity. That is, the duration and resource requirement amounts may vary between the modes of an activity. If multi-mode is considered rather than single mode, then this problem is called multi-mode RCPS, but single mode version of it will be shortly mentioned at first in this chapter.

There generally exist three types of resources (Slowinski [127]). Renewable resource is a limited resource in each period of the project. In other words, when the period finishes, the resource is renewed and becomes prepared for a new period. The example of it can be manpower. On the other hand, non-renewable resource is limited in total over the project duration. The example of it can be financial resource. An amount of money is available at the beginning of the project and as long as the project proceeds, this amount declines through the end of the project. Doubly-constrained resource is limited both for each period and for all project progress.

Mathematical formulation of the single mode RCPS is given as the following: Let  $x_{jt}$  equal to one if activity  $j$  completes in period  $t$ , otherwise it equals to zero. Let  $N$  denote the number of activities existing in the project.  $E_j$  and  $L_j$  denote the earliest and latest finish time of the activity  $j$ , respectively. These values are obtained by performing forward and backward recursion.  $P$  denotes the set of all pairs of immediate predecessor activities. That is,  $(a, b) \in P$  denotes that the activity  $a$  precedes the activity  $b$ . Let  $d_j$  denote the duration of activity  $j$ . Activity  $j$  requires the renewable resource  $k$  in the amount of  $r_{jk}$  and the capacity of this resource is denoted as  $R_k$ . Each activity uses the non-renewable resource  $i$  in the amount of  $w_{ji}$  and the capacity of this resource is represented by  $W_i$ .  $H$  denotes the known heuristic project completion time. Now, the mathematical formulation can be given:

$$\min \sum_{t=E_N}^{L_N} tx_{Nt} \quad (1.1)$$

$$\text{subject to } \sum_{t=E_j}^{L_j} x_{jt} = 1 \quad \forall j \in [1, N], \quad (1.2)$$

$$- \sum_{t=E_a}^{L_a} x_{at} + \sum_{t=E_b}^{L_b} (t - d_b)x_{bt} \geq 0 \quad \forall (a, b) \in P, \quad (1.3)$$

$$\sum_{j=1}^N \sum_{q=t}^{t+d_j-1} r_{jk} x_{jq} \leq R_{kt} \quad \forall k \in [1, K] \text{ and } \forall q \in [1, H], \quad (1.4)$$

$$\sum_{j=1}^N \sum_{t=E_j}^{L_j} w_{ji} x_{jt} \leq W_i \quad \forall i \in [1, l], \quad (1.5)$$

By putting (1.1), this model is forced to minimize the completion time of the last activity which represents the Cmax. With (1.2), an activity can not have finishing times more than one. (1.3) maintain the predecessor constraints. The last two constraints (1.4) and (1.5) consider the resource limits.

In addition to this formulation which is for only single mode RCPSP, multi-mode version of this problem is formulated as below:

- $x_{jmt}$ : if activity  $j$  operating in mode  $m$  completes in period  $t$ , otherwise it equals to zero.
- $d_{jm}$ : duration of the activity  $j$  operating in mode  $m$ .
- $r_{jkm}$ : the amount of renewable resource  $k$  required to operate the activity  $j$  in mode  $m$ .
- $w_{jim}$ : the amount of non-renewable resource  $i$  required to operate the activity  $j$  in mode  $m$ .

Besides these parameters, the earliest and latest finish time of the activities denoted by  $E_j$  and  $L_j$  are the same, with the only difference that forward and backward recursion are operated with the smallest duration mode. At this point, the multi-mode formulation of RCPSP can be given below:

$$\min \sum_{m=1}^{M_N} \sum_{t=E_N}^{L_N} t x_{Ntm} \quad (1.6)$$

$$\text{subject to } \sum_{m=1}^{M_j} \sum_{t=E_j}^{L_j} x_{jtm} = 1 \quad \forall j \in [1, N], \quad (1.7)$$

$$- \sum_{m=1}^{M_a} \sum_{t=E_a}^{L_a} x_{atm} + \sum_{m=1}^{M_b} \sum_{t=E_b}^{L_b} (t - d_{bm}) x_{btm} \geq 0 \quad \forall (a, b) \in P, \quad (1.8)$$

$$\sum_{j=1}^N \sum_{m=1}^{M_j} \sum_{q=t}^{t+d_j-1} r_{jkm} x_{jqm} \leq R_{kt} \quad \forall k \in [1, K] \text{ and } \forall q \in [1, H], \quad (1.9)$$

$$\sum_{j=1}^N \sum_{m=1}^{M_j} \sum_{t=E_j}^{L_j} w_{jim} x_{jtm} \leq W_i \quad \forall i \in [1, l], \quad (1.10)$$

It should be noted that network representation in this thesis is based on activity-on-node representation. In other words, activity is represented on the node and precedence relations between the activities are shown with the arcs.

RCPSP has been studied in the literature both with single projects and multi-project. Firstly, the studies related to single objective RCPSP are explained. Secondly, the literature about multi-project RCPSP is given in Section 2.3 of Chapter 2.

As can be seen in the literature, exact, heuristic and metaheuristic solution methods have been developed for solving RCPSP. In this thesis, metaheuristic solution methods are focused and among them, genetic algorithm (GA) is taken into account. More specifically, we aim at developing multi-objective GA-based solution method for effectively and efficiently solving RCPSP.

In this thesis, our motivation is to deal with multi-project RCPSP because this problem is real-life problem. Many companies, or the departments of the companies encounter enormous number of situations in which they should schedule the activities of the multi-projects in order find good solutions. Thus, they should consider many projects simultaneously. Moreover, in recent years, the companies have been forced to take into account many objectives simultaneously because of the increase of the system complexities and competition between the companies. All of these issues motivate us to study this problem.

The outline of the thesis is as follows: Chapter 2 presents the literature in detail. In Chapter 3, solution approach is explained elaborately. The next chapter, Chapter 4, presents the literature about fine-tuning procedure and describes the fine-tuning procedure

for both single project and multi-project case applied in this thesis. Chapter 5 proposes some divergence applications and improvement procedures. Computational studies are presented in Chapter 6. The last chapter, Chapter 7, concludes the thesis.

# Chapter 2

## Literature Survey

### 2.1 Single objective

#### 2.1.1 Exact solution methods

##### 2.1.1.1 Minimization of Cmax

- *Single mode case:* Brucker et al. [19] solve RCPSP with the minimization of Cmax. Preemption is not allowed and each activity uses constant amount of renewable resources. Additionally, precedence relations between activities are taken into account. Branch and bound algorithm (B&B) is proposed for solving this problem and the test problems developed by Kolisch and Sprecher [89] are used for evaluating the algorithm proposed in the study.

Demeulemeester and Herroelen [42] solve RCPSP. In this study, B&B algorithm is used with depth first solution strategy. Preemption is not allowed and multiple renewable resources are required for activities to proceed. The amount of resources required by an activity per period does not change. The objective is the minimization of total project completion time with satisfying the resource constraints and precedence relations between activities.

Demeulemeester and Herroelen [43] report the new insights obtained by using the modified version of B&B procedure proposed by Demeulemeester and Herroelen [42]. They try depth first, best first and hybrid strategy on this algorithm. Additionally, they define stronger lower bound. The problem characteristics remain the same with the former study.

Mingozzi et al. [101] focus on RCPSP. Activities require constant amount of renewable resources in each period. In addition, there exist precedence relations between activities that have to be obeyed. The objective is the minimization of  $C_{max}$  without violating the resource constraints and precedence relations. They give classical mathematical formulation which leads to lower bound calculations in the literature until that time. Afterwards, the authors define a new mathematical formulation which helps creating new lower bound calculation methods. B&B procedure using newly developed lower bounds and depending on newly proposed mathematical formulation is proposed.

Bartusch et al. [9] study RCPSP with the minimization of  $C_{max}$ . They consider that there may be time windows such as minimal and maximal time lags between activities. After proposing a structural approach, B&B algorithm exploiting this approach is developed.

De Reyck and Herroelen [36] focus on RCPSP with generalized precedence relations in order to minimize  $C_{max}$ . That is, there exist arbitrary minimal and maximal time lags between activities. The proposed method which is depth first B&B algorithm is appropriate for general class of scheduling problems.

Fest et al. [52] solve RCPSP with the objective of  $C_{max}$  minimization and minimal or maximal time lags exist between activities. The authors devise a new resource conflicts method when employing their B&B algorithm. This modified method has some advantages and drawbacks when compared to other B&B algorithms.

Dorndorf et al. [45] study RCPSP with generalized precedence constraints while minimizing  $C_{max}$ . The proposed method is time-oriented B&B method which depends on constraint propagation technique and uses temporal and resource constraints of the problem. It is stated that this solution method can be applied to other regular objectives. Additionally, truncated version B&B algorithm is compared to other heuristic methods.

Brucker and Knust [18] develop a new lower bound calculation method for RCPSP in order to minimize  $C_{max}$ . Activities have finish to start precedence relations and require certain amount of resources. The lower bound calculations contain two methods, one of which uses constraint propagation technique and the other one uses linear programming formulation. An algorithm containing these two approaches is proposed by the authors.

• *Multi-mode case:* De Reyck and Herroelen [38] focus on RCPSP with three significant properties. Generalized precedence constraints exist between activities, activities require multiple renewable, non-renewable and doubly-constrained resources and activities can be performed in different ways; that is, they can have several execution modes. The authors propose local search-based solution methodology and tabu search (TS) pro-

cedure. The problem is divided into two sub-problems: Mode assignment phase and scheduling phase with assigned modes. It is explained that these sub-problems are both  $\mathcal{NP}$ -hard. Thus, multi-mode RCPSP should also be  $\mathcal{NP}$ -hard.

Sprecher et al. [129] solve RCPSP optimally. They consider that each activity has multiple modes. B&B algorithm is proposed by the authors. Several bounding rules are presented. In one of them, preprocessing operation is performed. The detail of this procedure is explained in Chapter 3.

### **2.1.1.2 Maximization of net present value**

De Reyck and Herroelen [37] solve RCPSP with discounted cash flows and generalized precedence relations. In other words, it is an extended version of RCPSP with minimal and maximal time lags between activities and with the objective of maximizing the net present value (Npv) of the project calculated by taking into consideration positive or negative cash flows for each activity. The authors propose depth first B&B algorithm in order to solve this problem.

Vanhoucke et al. [144] solve RCPSP with the maximization of Npv. Activities have constant amount of resource requirements and positive or negative cash flows. At the end of execution of activities, progress payments and cash outflows are realized. Depth first B&B algorithm is proposed by the authors.

Doersch and Patterson [44] consider the project scheduling problem with the maximization of Npv. Objective function includes activity cash flows and any penalties resulting from delayed completion of the project. In order to solve the problem, zero-one integer programming model has been developed. The authors claim that the model proposed in the study solves the problem optimally. However, for larger data sets, it can not reach to an optimal solution.

Icmeli and Erenguc [73] focus on RCPSP. Although the most frequently considered objective is the minimization of Cmax, they deal with the maximization of Npv because this objective is more financial motivated and realistic. They propose B&B procedure to solve this problem. Resource violations that might be encountered are tried to be solved by adding additional precedence relations between certain activities. This method is called minimal delaying alternatives. The bounds are computed by solving Payment Scheduling Problem.

Kazaz and Sepil [77] deal with project scheduling problem the objective being the maximization of Npv. In that study, contrary to tradition, cash inflows are assumed to occur at the end of a period such as one month, one year; which is called progress pay-



ment. However, cash outflows are realized at the end of corresponding activities. The authors present the mixed-integer programming formulation of this problem. For solving the problem, Benders Decomposition technique is proposed and significant computational results are observed.

### **2.1.1.3 Other objectives**

Vanhoucke et al. [143] consider RCPSP with the minimization of weighted earliness-tardiness penalty cost. Activities have a known due date and constant amount of renewable resource requirement. The authors propose depth first B&B algorithm. Finish to start precedence relations exist between the activities.

Drexl [46] consider RCPSP as assignment type problem. For this problem, a stochastic scheduling heuristic and a hybrid B&B/dynamic algorithm have been presented. Minimization of the total cost is the unique objective function in this problem. The problem presented in the study considers precedence relations between activities, resource-resource trade-offs and time-resource trade-offs. Additionally, each job has a release date and a deadline.

### **2.1.1.4 Other problems**

Slowinski [127] considers the Resource Allocation Problem which assigns the resources to the activities by considering the resource capacities and trying to improve the objective. Each activity has three type of resources; renewable, non-renewable and doubly-constrained. Two different solution methods are proposed by the authors. Those are both dependent on the linear programming formulation of this problem.

## **2.1.2 Heuristic solution methods**

### **2.1.2.1 Priority rules**

#### **2.1.2.1.1 Minimization of Cmax**

Davis and Patterson [34] solve RCPSP with the objective of project duration minimization. They compare eight different priority-rule heuristics, proposed in order to solve this problem, relative to an optimal solution. Those priority rules are minimum job slack (MINSLK), resource scheduling method (RSM), minimum late finish time (LFT), greatest resource demand (GRT), greatest resource utilization (GRU), shortest imminent

operation (SIO), most jobs possible (MJP) and select jobs randomly (RAN). It is observed that MINSLK is the best heuristic among others and RSM and LFT are close to MINSLK in terms performance. Thus, these three heuristics outperform other methods.

Kolisch [85] deals with RCPSP. Specifically, he focuses on RSM priority rule and analyzes it. It is observed that it leads to poor solutions. Thus, an improved version of it is developed and two new priority rules, which depends on MINSLK rule and resource based slack priority rule are proposed by the author. With the help of experimental investigation, those newly proposed method are observed to outperform other priority rules developed until that time.

Kolisch [86] studies RCPSP with the objective of total project completion time minimization. Specifically, he focuses on the comparison of serial and parallel schedule generation schemes (SGS) in depth. The author refers to some studies to show which exact and heuristic methods have been studied. Among heuristic methods, it is said that priority rule based scheduling heuristic are the most important procedure. In each stage of this method, eligible set of activities (that is, activities whose all predecessors have already been scheduled) is found and one of these activities which is selected by using given priority rule is scheduled. For scheduling, two different methods (serial and parallel SGS) are used.

SGS defined so far is termed as single pass approach, which means one priority rule and one generation scheme are used and only one solution is obtained. However, in multi-pass approach,  $Z$  single passes are applied so as to provide a sample of at most  $Z$  unique feasible solutions whose the best solution is selected. Two different kinds of multi-pass approach can be explained. Whereas multi-priority rule method uses one scheduling scheme and various priority rules, sampling benefits from one scheduling scheme and one priority rule.

The author gives some observation about generation schemes. He states that both of them can obtain feasible schedules with respect to both precedence relations and resource availabilities. Secondly, it is said that both methods reduce to a simple forward recursion in resource unconstrained case. Additionally, he suggests a theorem stating that serial SGS obtains active schedules and parallel SGS provides non-delay schedules.

Ulusoy and Ozdamar [136] solve RCPSP with a priority rule developed in that study. The objective is the minimization of total project duration. Types of network and resource characteristics are investigated in order to determine whether they affect the priority rules' performance. The authors develop a new priority rule called weighted resource utilization and precedence (WRUP) and it is observed to outperform other most commonly used

priority rules such as MINSLK, LFT and RSM.

Khattab and Choobineh [78] solve the single resource RCPSP. In addition to existing ten priority rules in the literature that can solve this problem heuristically, they define new eight priority rules. The method taking into priority rules consideration to solve this problem ranks the activities with respect to corresponding priority rule and then schedules them in earliest resource feasible time so that the Cmax of the project is minimized. By considering these priority rules, the authors propose a new heuristic called SEARCH working with those rules and performance of it is compared to other heuristics.

Some performance measures are defined to compare the newly proposed priority rules with already existing ones. In that paper, five different performance measures are defined: Resource utilization measures how well a schedule uses its resources as a function of time. Average deviation from the shortest known duration is the difference between the best known project duration and the project duration obtained by the  $k^{th}$  heuristic averaged over all projects. Another measure is the frequency of obtaining the shortest duration considering how many times a heuristic obtains the best duration when compared to other heuristics. Resource range is another one measuring the difference between the minimum resource level required to complete the project and the resource level needed to schedule the activities by critical path method. The last measure is project delay which calculates the difference between the project duration computed by assigning the resources which have minimum usage levels to the activities and the project duration computed with critical path method.

Boctor [12] deals with RCPSP with the objective of minimizing Cmax. The authors proposed some multi-heuristic methods containing both parallel and serial rules. The performance measure used in the study is either the percentage of critical path length (for large problems) or optimal solution in the case that it is known.

Neumann and Zhan [109] solve the problem of the minimizing the total project duration in the presence of resource constraints where there may be minimal and maximal time lags between activities. The authors propose efficient priority rule heuristics for solving this problem.

#### **2.1.2.1.2 Maximization of net present value**

Selle and Zimmermann [123] consider RCPSP with the objective of Npv maximization. Temporal constraints exist between the activities. The authors propose bidirectional priority rule based method. Well known serial SGS is compared to this proposed algo-

rithm and the efficiency of the latter algorithm is shown.

Neumann and Zimmermann [110] focus on RCPSP by considering two different objectives. When they try to minimize the resource usage deviation, they call this problem as resource leveling. Otherwise, if they consider the financial aspect of the project, they call this problem as Npv problem. Minimal and maximal time lags exist between the activities. They construct that study by considering various aspect of both problems such as whether there is resource constraint and whether minimal or maximal or both of them are used in the problems. The authors propose exact and heuristic methods (a different TS algorithm) for resource leveling for constrained and unconstrained case and Npv problem for unconstrained case. In addition, a heuristic procedure is developed for Npv problem when there exist resource constraints.

Sepil and Ortac [124] focus on RCPSP with the aim of the maximizing Npv. Cash inflows are not defined to occur at event realization times, rather they are defined to occur periodically. In that case, cash inflows do not depend on the activities. Single-pass greedy forward algorithm is proposed to solve this problem and three different priority rules are included in this algorithm. The aim of these rules is to select an activity to be scheduled from the eligible activities.

### **2.1.2.2 Forward and backward recursion**

Li and Willis [98] solve RCPSP with forward and backward recursion. This procedure continues scheduling the activities until there is no improvement in the objective function, which is the minimization of project completion time. It is stated that this method leads to cheap and short schedule because backward recursion tries to schedule the activities as late as possible.

Ozdamar and Ulusoy [115] deal with RCPSP with the minimization of Cmax. They propose iterative forward and backward scheduling technique, which is based on the study Li and Willis [98] to solve this problem. However, those techniques are different in the algorithmic way and the activity selection method.

### **2.1.2.3 Sampling method**

Cooper [32] deals with RCPSP with the minimization of Cmax. He studies two groups of heuristic methods, both of which use priority rules. First one is parallel method generating one schedule and the second one is sampling method generating multiple schedules and selecting the best one. The impacts of the heuristic methods, network characteristics and

priority rules are evaluated.

#### **2.1.2.4 Variable neighbourhood search**

Fleszar and Hindi [54] solve RCPSp with variable neighbourhood search, which is a heuristic solution method proposed by Mladenović and Hansen [102]. The solution is coded with activity list that does not violate precedence relations between the activities. The activity list is turned into the schedule by using serial schedule generation scheme. In this study, the solution space is explored by utilizing effective lower bounds and precedence augmentation. The computational study is conducted with 2040 benchmark instances. According to the result, the best known solutions are improved for some instances. The authors say that variable neighbourhood search is inferior for some instances, but even in this case, average deviation is small.

Bouffard and Ferland [14] create another solution method consisting of simulated annealing (SA) and variable neighbourhood search to solve resource constrained scheduling problem. Actually, they try to improve SA procedure with variable neighbourhood search. The author compare this method to other local search methods, which are threshold accepting methods and tabu search. These methods are combined with multi-start diversification strategies and with the variable neighborhood search technique. A detailed computational study is performed to evaluate the performances of these methods. According to the results, simulated annealing improved with variable neighbourhood search outperforms the other methods.

#### **2.1.2.5 Other methods**

Brinkmann and Neumann [16] focus on the problem of leveling the resource consumption and minimizing the total project completion time. Minimal and maximal time lags are allowed to exist between starting time of the activities. The authors propose two different heuristics called direct method and contraction method. At the end of the empirical analysis, the contraction method becomes superior over the other method for minimum completion time problem. For resource usage leveling problem, both methods are seen to behave similarly.

Cesta et al. [22] solve RCPSp with generalized precedence relations. They propose a heuristic method, which depends on the constraint satisfaction problem (CSP) solving search procedure. This procedure tries to find "resource contention peaks" (that is, the set of activities whose resource requirement summations are the maximum) and the levels.

Naphade et al. [108] focus on RCPSP by proposing two heuristics based on Problem Space Search metaheuristic procedure. This algorithm is compared to B&B method developed by Demeulemeester and Herroelen [42]. It is observed that proposed heuristics perform well when comparing to B&B algorithm.

Sampson and Weiss [121] deal with RCPSP by applying local search technique. A solution representation is determined so that it can be precedence feasible. It is explained that two significant advantages of local search technique are its ability to deal with arbitrary objective functions and its efficiency over a wide range of problem sizes. The proposed method is observed to have important improvements over the best heuristics developed until that time.

Ulusoy and Ozdamar [137] deal with RCPSP with the objective of the minimization of  $C_{max}$ . A new heuristic method called local constrained based analysis (LCBA) is proposed and it is said to be more robust than the dispatching rules existing in the literature. After computational studies, it is observed that LCBA outperforms dispatching rules and can obtain near optimal time efficient solutions.

#### **2.1.2.6 Other objectives**

Yamashita et al. [150] focus on RCPSP to minimize resource availability cost. The authors call this problem Resource Availability Cost Problem (RACP). They claim that several exact methods for solving it have been proposed, but they are aware of heuristic methods. Thus, a heuristic method called scatter search has been developed. For small instances, scatter search has been compared to known exact methods, but multi-start heuristics has been proposed in order to show the quality gain generated by scatter search for medium and large instances.

### **2.1.3 Metaheuristic solution methods**

#### **2.1.3.1 Genetic algorithm**

##### **2.1.3.1.1 Minimization of $C_{max}$**

- *Single mode case:* Hartmann [64] focuses on RCPSP with single objective. Individuals are constructed as precedence feasible activity list (the other name is permutation based genetic encoding). While forming the initial population, one of the unscheduled activities whose predecessors have already been scheduled is selected and put in the or-

der. In addition, a sampling procedure is used in order to form the activity list. That is, one activity is selected for the next position with a probability derived by a priority rule.

Crossover is applied in three different ways. One-point crossover and two-point crossover will be explained in Chapter 3. The third method called Uniform Crossover generates  $N$  (the number of activities) random integer. However, these random number can take only two values, 0 and 1. While creating the daughter, if  $N_t$  is equal to 0, then that activity existing in  $t^{th}$  position is taken from the mother. If not, that is taken from the father. In this type of crossover, algorithm is designed so that precedence relations can be taken into account.

Mutation operator considers all activities. The activities  $j_t$  and  $j_{t+1}$  are changed with a probability  $p_m$ , if the resulting sequence does not violate the precedence relations.

For selection operator, four different ways are defined: Ranking method, proportional selection, where each individual has a probability of dying, 2-tournament and 3-tournament selection.

Priority value based GA has been considered by the author. In this representation, individuals are presented with randomly generated priority values. Crossover methods are the same with those of precedence feasible representation. Mutation is performed as follows: For all activities, a new random priority value is generated with a probability  $p_m$ .

Priority rule based GA has also been paid attention by the author. For each position of the individual, a priority rule exists to schedule an activity. Crossover and selection operators are the same. However, mutation operator is performed by generating a new priority rule for each position with a probability  $p_m$ .

Leon and Balakrishnan [97] deal with RCPSP with considering two objectives separately; minimization of Cmax and mean tardiness. They propose a method that can be regarded as local search making use of problem-space based neighborhoods. This method can be beneficial for all regular objectives, if it is adapted. By using GA additionally, further improvements are observed. Close-to-optimal solutions are provided in both cases; with and without GA.

Alcaraz and Maroto [4] solve RCPSP with single mode. Objective adopted in that paper is the minimization of Cmax. The solution is tried to be reached by GA. However, they propose new solution representation and crossover operators. Thus, it turns out to be that a new GA is proposed by the authors. Basically, various changes in genetic operators such as population size, crossover rate and mutation rate have been observed and compared each other.

They present different solution encoding techniques. Activity list representation con-

constructs the feasible list. In other words, every activity comes after its all predecessors in this list. Parallel SGS can not be applied to this type of representation. The authors give the list of studies that have used this kind of representation. Priority rule representation include some rules to schedule the activities. The rule  $P$  existing in  $t^{th}$  position of activity list has to be used while determining  $t^{th}$  activity of the schedule. Parallel and serial SGS can be used for scheduling the activities. Random key representation and shift vector representation has been also presented by the authors. Finally, Alcaraz and Maroto [4] have preferred the activity list representation. However, they added a gene to the list indicating the method by which the schedule is constructed. This gene determines whether this schedule is constructed by forward or backward schedule.

During forming the initial population, activity list is constructed with the help of LFT rule. Thus, the authors have use sampling approach for computing the initial population. After solving the schedules with two different methods (forward and backward), the algorithm selects the one with the smallest Cmax.

For selection mechanism, three methods have been presented. The remainder stochastic sampling without replacement, 2-tournament selection and ranking method have been presented. For crossover operator, firstly the checking method about whether crossover is beneficial or not is proposed. Afterwards, three types of crossover are proposed. Precedence set crossover is applicable for standard activity list. Forward and backward crossover technique is beneficial for newly proposed solution encoding. In this operator, the last gene of activity list becomes important because it affects the crossover procedure. Two-point forward backward crossover is also applicable to newly proposed activity list. Thus, it takes into the last gene consideration during crossover. The only difference between the previous one and two-point crossover is that two random integers have been used for the latter one.

In order to perform mutation operator, two kinds of this operator have been used. One of them is to select random position for each activity by preserving the precedence feasibility and put this activity in this randomly selected position. In the second one, each position is swapped with the following one with a probability. If the last gene is for scheduling method, then this gene is changed with a probability during mutation procedure.

- *Multi-mode case:* Mori and Tseng [104] solve RCPSP with multi-mode. Cmax is aimed to be minimized. The authors define some heuristic methods which try to solve this problem and they develop a new heuristic method GA. Deterministic scheduling rules proposed by Talbot [132] and stochastic scheduling method defined by Drexl and Grnewald



[47] are selected to be compared to GA.

#### **2.1.3.1.2 Maximization of net present value**

Ulusoy et al. [139] solve the single objective form of RCPSP with discounted cash flows. This type of problem is generally represented as RCPSPDCF. They apply four different payment models on this problem; lump sum payment, payments at event occurrences, equal time intervals and progress payment. The problem is solved by GA. Initial population is constructed randomly. For each individual, one of the activities whose predecessors are already in the scheduled list is put in this list. Schedule generation is performed by scheduling each activity in resource and precedence feasible time. Additionally, the starting time of an activity can not be larger than the starting time of the previous activity in the feasible list. If an individual is infeasible with respect to resources, it is discarded. As for crossover and mutation, if crossover is applied, then two parents from current population are selected and one child is produced by crossover. Otherwise, fitness-based selection is applied to the current population to select a chromosome to be reproduced as the offspring. Then, this child is applied mutation operator. Of course, this newly generated child is checked if it is feasible with regard to resources. If not, it is discarded. Chromosome representation is succeeded by activity list with mode assignment. The crossover operator is multi-component uniform order-based crossover operator.

Ulusoy and Cebelli [135] solve RCPSP to maximize Npv. They propose double-loop GA for solving this problem. In double-loop representation of the problem, outer loop and inner loop represent the client and the contractor, respectively. Information flows between client and contractor. More specifically, the knowledge of timing of payment becomes available for the client and the knowledge of payments amount becomes available for the contractor.

#### **2.1.3.1.3 Other formulations**

Shadrokh and Kianfar [125] solve Resource Investment Problem which is a branch of RCPSP. The objective is the minimization of cost resulting from activities and overdue of the project. Thus, a cost is defined for each activity and a due date is given for the project completion. For solving the problem, GA is proposed.

Crossover operator is performed with a probability  $p_c$ . If this probability is not satisfied, then each of the selected parents are applied only mutation operator and local search.

If satisfied, then the algorithm decides whether the generated individuals should exist in the population or be discarded by a comparison operator. After constructing the population, immigration operator is applied to this population. A new randomly generated individual is compared to randomly selected individual from the population. If new generated one wins the other one, it replaces the selected one in the population.

Chromosome structure is the precedence feasible activity list. Capacity list exists with the activity list. To compute the fitness value, the equation calculates the cost of every activity and tardiness cost if the schedule is overdue.

Crossover is performed in three types. One-point and two-point crossover by Hartmann [64] are performed. The other operator is type-2 two-point crossover. For the capacity list, a different crossover technique is applied. Mutation is performed as follows: For the activity  $\alpha$ , let the last predecessor of it be activity  $\beta$  and the first successor of it be  $\theta$ . Random integers  $x \in [\beta + 1, \theta - 1]$  and  $y \in [0, N]$  is generated, where  $N$  is the number of activities. Afterwards, activity  $y$  is inserted between the activities  $x - 1$  and  $x + 1$ .

Najafi and Niaki [107] focus Resource Investment Problem (RIF). This problem is defined in the literature that resource availability levels are decision variables. In this problem, there is a deadline for project completion time. Since the costs are realized for providing the resources, the objective is to find a schedule and resource levels such that objective function can be minimized.

The authors propose GA to solve this problem. Initial population is created randomly. For generating the next population, the best individuals determined by evaluating the fitness functions of the individuals are copied to the next generation. Afterwards, the population is divided into two parts randomly. For each pair of individuals, crossover operator is applied with a probability. Then, two obtained individuals are subject to mutation operator with a probability. Finally, a local improvement procedure is executed so as to improve the provided schedules with a probability.

Chromosome representation is accomplished through feasible activity list. Each gene in the chromosome denotes the floating time of an activity in that schedule. That is, floating time  $F_H(i)$  of activity  $i$  at schedule  $H$  is equal to  $S_H(i) - ES_i$ , where  $S_H(i)$  denotes the starting time of the activity  $i$  at schedule  $H$  and  $ES_i$  denotes the earliest starting time of activity  $i$  obtained with critical path method.

For creating initial population, two methods are used with equal probability. These are forward and backward approach for chromosome representation. Crossover is defined as specific for the chromosome representation. For each crossover execution, a random integer number between  $[1, N - 1]$  is generated, where  $N$  denotes the number of activities.

Then, the parents are divided into two parts with respect to this randomly generated number. Afterwards, backward and forward floating time of the activities are calculated and crossover is performed with these values. For mutation operator, two integer values are generated between  $[1, N - 1]$ . For the selected intervals with respect to these values, the backward or forward floating time of each activity in this interval is calculated randomly.

Finally, two different local improvement techniques are proposed by the authors. First one is the removal of the empty period at the schedule. Second one is achieved through the shifting of activities either to left or to right depending on whether the activity has forward floating time or backward floating time.

### **2.1.3.2 Simulated annealing**

Cho and Kim [28] study RCPSP with the objective of Cmax minimization. They propose SA algorithm and a solution is represented with priority list. To generate schedules, a priority scheduling method is performed. In the proposed algorithm, some activities can be delayed so as to have larger solution space due to the fact that non-delay schedules can not always contain an optimal schedule.

Pan and Yeh [117] focus on RCPSP. They think that the knowledge about activity parameters (in particular, the duration of the activities) is not precise. Thus, fuzzy set theory is used by the authors since they think that it is the best way to deal with such a situation. In order to solve the problem, fuzzy SA approach is proposed. The unique objective is the minimization of total project completion time.

Probabilistic way is used for handling imprecise cases about activities, but although it is valid in theory, it is not efficient method in practice since project managers generally have not information about the parameters in advance. Thus, fuzzy set theory is regarded as the best way to handle this problem.

### **2.1.3.3 Tabu search**

Icmeli and Erenguc [72] study RCPSP with discounted cash flows. Each activity possess a cash inflow or outflow and the objective is the maximization of Npv of the project. A tabu search (TS) procedure is proposed by the authors and this method is modified to call long term memory function. It is observed that this method can produce near optimal solutions within reasonable time.

#### **2.1.3.4 Other methods called hybrid**

Lee and Kim [96] consider RCPSP to minimize  $C_{max}$ . Authors prefer to use three search heuristics, which are TS, SA and GA. The solution is represented by a set of numbers, each of which denotes the priority value for the corresponding activity. It is said that this representing method can always give feasible neighborhood solutions. These search procedures are compared to already existing heuristics in the literature; minimum slack method, the iterative method and the SEARCH method proposed by Khattab and Choobineh [78].

In GA phase, one-point crossover method is applied. As for mutation operator, two genes are selected randomly from the string and their values are exchanged with a probability.

#### **2.1.4 Review papers**

Herroelen et al. [69] review the project scheduling problem with the objective of Npv maximization. Firstly, they define Npv criterion and explain contracts and payment structures. Various problem types and assumptions (changing with objectives, resources, etc.) are given by the authors in detail.

Hartmann and Kolisch [67] evaluate the state-of-the-art heuristics for RCPSP. They consider this problem with single objective, which is the minimization of  $C_{max}$ . At first, they explain SGS in detail. They point out that serial SGS always holds optimal schedule in the set of active schedules. It generates optimal active schedules for unconstrained resource case. List scheduling is a variant of serial SGS. On the other hand, parallel SGS can be regarded as alternative method for serial SGS. Parallel SGS generates non-delay schedules, which may be optimal for unconstrained resource case. However, this has a drawback that it may exclude the optimal schedules.

They define X-pass methods. They contain single-pass method and multi-pass method. Afterwards, the authors show the most important metaheuristics for RCPSP, which are TS, SA and GA. They have tested various problem properties, different initial population constructing techniques and different SGSs. Furthermore, different metaheuristics have also been tested.

Finally, it has been observed that the most successful metaheuristics are SA procedure of Bouleimen and Lecocq [15] and GA of Hartmann [64]. Activity list representation outperforms the best among other representation techniques. Furthermore, serial SGS can be seen to be better than parallel SGS, but in larger problems, the latter one can be

better than the former one.

Kolisch and Hartmann [88] evaluate the most recent papers and heuristic methods proposed since the last survey of Hartmann and Kolisch [67]. In the last survey, the most promising heuristics were GA of Hartmann [64] and SA procedure of Bouleimen and Lecocq [15]. Some heuristics developed since the last survey seem to outperform them. The heuristics developed by Alcaraz et al. [5], Debels et al. [40], Hartmann [66], Kochetov and Stolyar [83] and V. et al. [140] are tested and accepted as the most successful heuristic developed for RCPSP. In addition, it is said that forward and backward improvement is one of the most significant component in four of these six important heuristics.

## **2.2 Multi-objective**

### **2.2.1 Exact solution methods**

Nudtasomboon and Randhawa [113] study RCPSP. Single objective versions of this problem are evaluated before and some algorithms are proposed. The authors examine the minimization of  $C_{max}$ , the minimization of project total cost and the minimization of variation in resource levels. Zero-one integer programming model is developed for this problem. As for solution algorithms, extensions of the implicit enumeration technique are used to develop these algorithms. Afterwards, zero-one preemptive goal programming approach is proposed for the multi-objective case of RCPSP. These objectives include time, cost and resource leveling aspects. In that study, several components of RCPSP are described and included in the model proposed throughout the paper. These components are splittable / non-splittable jobs, three types of resources (renewable, non-renewable and doubly-constrained), variance in resource consumption and time-cost trade-offs.

Azaron et al. [7] deal with the time-cost trade-off problem. Each activity has the duration which is non-increasing function of the resource amount assigned to it. The authors try to solve this problem by using optimal control theory in Markov PERT networks. They develop multi-objective control problem and two different solution techniques called goal attainment and goal programming are used. The objectives are the minimization of total direct costs, the mean of project completion time and the variance of the project completion time.

### 2.2.2 Heuristic methods

Al-Fawzana and Haouari [3] study robustness maximization and total project time minimization simultaneously for the single mode RCPSP. They define robustness as the "project's ability to cope with small increases in the time duration of some activities that may result from uncontrollable factors". It is said that multi-objective combinatorial optimization problem can be solved by three types of algorithm. First, exact methods such as B&B and dynamic programming can be used. Second, metaheuristic methods such as GA, TS and SA can be beneficial in order to find the approximate efficient solutions. Third, some decision support system (DSS) with interactive usage can be developed. The DSS can include both exact methods and heuristic methods. In that paper, TS has been preferred. Two important performance measures for evaluating the quality of the obtained non-dominated solutions are proposed in this study.

Abbasi et al. [1] study the bi-objective case of RCPSP. The objectives are Cmax minimization and robustness maximization. They define robustness as "if the duration of an activity becomes larger than the estimated duration, the project completion time will not change without any cost". Objective function is formed as a weighted combination of those two objectives. In other words, they are combined into one objective function by assigning weight for each objective. SA has been performed to solve this problem.

Ulusoy and Ozdamar [138] study RCPSP and try to solve it with two objectives, the minimization of Cmax and the maximization of Npv. Two different cash structures are examined. In the first one, each activity has cash outflow at its start time and lump sum payment takes place at the project completion time. In the second structure, activities are allowed to have multi-mode. The authors apply an iterative scheduling algorithm for solving this problem.

Davis et al. [35] examine RCPSP as a multi-objective problem. They consider the minimization of project completion time and balancing the resource usage simultaneously. The method that is used within the study is an interactive procedure based on vector maximization.

Viana and de Sousa [146] focus on RCPSP with multiple objectives. They are the minimization of Cmax, the minimization of mean weighted lateness of activities and the violation of resource constraints. Although in the mathematical formulation, there are two constraints regarding the feasibility of resource usages for both renewable and non-renewable, the relaxation of these constraints leads to the objective of minimization of resource usage over capacity because the authors think that the constraints with respect

to resource usage may be violated in practical. Since the single objective form of this problem is  $\mathcal{NP}$ -hard, multi-objective case becomes even more difficult. It is stated that classical methods such as  $\varepsilon$ -method proposed for solving multi-objective form are not efficient by examining the study of Steuer [130]. Thus, multi-objective metaheuristics have been preferred in the study of Viana and de Sousa [146]. Specifically, Pareto simulated annealing (PSA) and multi-objective tabu search (MOTS) have been used for obtaining good approximation of non-dominated solutions. For evaluating the solution quality, two different performance metrics have been presented. First, the closeness of the obtained non-dominated set to true Pareto front is evaluated. For doing this, closeness measure defined in the literature is used. Secondly, the measure of distance to Zeleny point is utilized for evaluating the solution quality.

Hapke and R. [62] study RCPSP with renewable, non-renewable and doubly-constrained resources. The problem has three objectives considering the time and cost aspects. Taken from a real life problem, these objectives are accepted as the minimization of project completion time, handling of manpower resource smoothness and the minimization of total project cost. The problem studied by the authors has been considered by observing agricultural procedure. In this problem, solution procedure has two stages. In the first stage, PSA tries to find good approximate solutions to the problem in a reasonable time. Afterwards, Light Beam Search is executed interactively in order to examine the non-dominated solutions provided by the first stage.

Slowinski et al. [128] propose a DSS for multi-objective project scheduling. Renewable, non-renewable and doubly-constrained resources are included in the problem formulation. The objectives considered in that paper have time and cost aspects. In order to solve the problem, three heuristics are involved into DSS. These are parallel priority rules, SA and B&B procedure. In interactive phase, decision maker can have the algorithm to search for the best compromise schedule.

Ballestín and Blanco [8] study the multi-objective RCPSP. One important factor of this problem is that it has regular objective functions (Cmax, total tardiness, sum of start times, maximum tardiness and Npv with only positive cash flows). Some significant observations are given about multi-objective project scheduling problem (PSP) and multi-objective RCPSP. With the help of them, some heuristics have been proposed. Strength Pareto evolutionary algorithm (SPEA), non-dominated sorting genetic algorithm (NSGA) and PSA are those procedures.

In order to execute these algorithms, some details about the stages of procedures have to be shown. Firstly, two-point crossover and mutation of Hartmann [64] have been ap-

plied. The other significant component is that they use regret-based biased random sampling of Drexl [46] for generating the initial population. Additionally, sampling method combined with a local search called double justification can almost compete with the best metaheuristic algorithms. Moreover, different schedule generation schemes are tried. Although parallel SGS may not reach to an optimal solution, it uses the resources in the way that the resulting schedule is compact. That is, parallel SGS tries to use the resources in the sense that it leads to relatively small idle time. Thus, some authors like Hartmann [66] and Kochetov and Stolyar [83] have preferred to use combined form of serial SGS and parallel SGS. Finally, elitism has been studied to see whether it is effective in multi-objective RCPSP by comparing non-dominated sorting genetic algorithm - II (NSGA-II) with and without elitism.

Cochran et al. [29] study parallel machine scheduling problems. The objectives considered are the minimization of  $C_{max}$ , the minimization of total weighted completion times and the minimization of total weighted tardiness. The method preferred in that study is two-stage multi-population genetic algorithm (MPGA). Actually, the authors basically propose GA with dispatching rule, but MPGA has been presented in order to handle multi-objective case. MPGA is made up from two heuristics, which are vector evaluation genetic algorithm (VEGA) and multiobjective GA (MOGA). After developing the MPGA, this method is compared to MOGA, which is called the benchmark method with two objectives; the minimization of  $C_{max}$  and total weighted tardiness.

General scheme of MPGA has been presented by the authors. One important component is the selection mechanism. Each parent is selected with the probability. Crossover and mutation are performed in a simplest way. As for fitness function, it is formed as a single function by weighting the objectives. It is important that the weight of an objective in one iteration is generated randomly. Elitism is applied to this problem by preserving the best solution of each objective and the best solution of the combined objective. The procedure explained until here has been called the first stage. After obtaining the solutions of the first stage, the population is divided by rearranging the solutions with respect to each objective and combined objective. If there exist  $p$  objectives to be optimized, then the population is divided into  $p + 1$  sub-populations. The first  $p$  populations are for  $p$  objectives and the last one is for the combined objective. Selection, crossover and mutation operators of the second stage are the same with those of the first stage. In order to preserve the best solutions, algorithm searches the best solution for each objective over all populations. Thus, it provides  $p + 1$  best solutions to be preserved.

Slowinski et al. [128] study PSP with multiple category resources and propose a DSS



for solving this problem. Solution procedure takes into account several objectives simultaneously. In the model studied in that paper, renewable, non-renewable and doubly-constrained resources are included and activities are considered to have multi-mode. The structure of DSS is presented by the authors. First module of it is model editor. There are heuristic procedures, which solve the single objective PSP in the second module. These are parallel priority heuristics, SA and B&B method. The interactive procedure for solving multi-objective PSP is given in the third module. In this phase, the interactive module uses the solution obtained by one of the heuristics defined above. The aim of the fourth module is to display the results to the decision maker.

The objectives considered in that study are the minimization of project completion time, mean weighted lateness, total number of tardy activities, mean weighted flow time, weighted resource consumption and Npv. As a test problem, a hypothetically generated agricultural project consisting of 30 activities is used.

Hapke et al. [63] consider multi-mode RCPSP with time parameters of activities. Renewable, non-renewable and doubly-constrained resources are included in the model. The objectives are the project Cmax, resource utilization smoothness, maximum lateness, mean flow time, Npv and project cost. Since all time parameters of activities have fuzzy nature, the duration of a mode, ready time and due date of an activity have fuzzy property. Parallel SGS is used for scheduling the activities with some modifications to make it proper for the fuzzy nature of this problem.

Two-stage solution procedure is proposed for solving this problem. In the first stage, PSA is performed in order to obtain approximate non-dominated solution sets. In the second stage, an interactive procedure called Light Beam Search tries to find the best compromise solution over the non-dominated set with respect to the preference of the decision maker.

Nabrzyski and Węglarz [106] solve multi-mode RCPSP with multi-objective. These objectives are the project duration, mean weighted flow time, mean weighted lateness, weighted resource consumption, Npv and profit obtained in the project. The solution method has two phases. In the first stage, TS algorithm is used to solve this problem with a single objective and then, non-dominated solutions are searched by examining the single objective solutions. In the next phase, interactive procedure is performed so as to provide compromise solutions with regard to decision maker's preference.

Wang et al. [148] solve RCPSP with multiple objectives. The objectives are the minimization of total project time and satisfying of resource utilization of smoothness. The latter one is defined as the average deviation from the average resource usage. The pro-

posed method is NSGA-II. Chromosome representation is achieved with feasible precedence activity list and assigned mode for each activity. Serial SGS is preferred to schedule the activities. Initial population is created randomly. For selection operator, binary tournament is preferred. For daughter, crossover is applied as in the study of Hartmann [64], but the mode is carried without any change with the activity. With a probability, mutation is performed by selecting a position randomly, generating an activity and inserting this activity into the selected position if precedence relations are satisfied. For the mutation of modes, an activity is selected randomly and its mode is assigned randomly from its mode set. The proposed algorithm is tested on agricultural example taken from Hapke and R. [62]. It is seen that it can solve the problem efficiently.

Elloumi and Fortemps [49] study the multi-mode RCPSP with two objectives; minimization of  $C_{max}$  and satisfying the non-renewable resource feasibility. Although they approach to this problem as single objective, it is converted to bi-objective case since they want to solve the infeasibility of non-renewable resources at the other objective. They implement evolutionary algorithm to solve this problem, but with a new idea. They propose an adaptive grid relying on clustering technique in order to avoid premature converge of the algorithm.

Individual is represented by a pair of activity list and mode assignment list like in the study of Hartmann [65]. Preprocessing of the data file (that is, reduction of the search space) is performed as in the study of Sprecher et al. [129]. In other words, non-executable modes are removed first. Then, redundant non-renewable resources are eliminated and all inefficient modes are deleted. These last two operations are repeated in a loop until an improvement can not be provided. Initial population is constructed in the same way of Hartmann [65]. Since the algorithm proposed by the authors allows infeasible solutions with respect to non-renewable resources, an aggregate value defined for leftover capacity of non-renewable resources is accepted as one objective to be minimized. As for fitness function, the rank-based fitness assignment (see Section 3.6 in Chapter 3) method is used.

For density computation, they avoid to use the classical method called cell-based density approach because it has some shortcomings. Thus, they propose a new idea called clustering-based density approach. Crossover operator is applied in the same way of Hartmann [65]. Mutation operator is applied in two stages. In the first one, an activity  $j$  is selected and is changed with  $j + 1$  with a probability, if precedence relation is not violated. If not changed, then another activity is selected and is tried to be changed with neighbor activity. These procedure continues until an activity is changed or  $N$  (the number of activities) unsuccessful trials are executed. In the second stage, an activity is selected and

its mode is changed.

Ghoddousi et al. [56] study the multi-mode RCPSP with its extended version. Since multi-mode RCPSP, the discrete time-cost trade-off problem and the resource allocation and resource leveling problem are significant project scheduling problem, the authors consider all of them in one problem. To solve this problem, multi-mode RCPSP is changed to multi-mode resource-constrained discrete time cost-resource optimization model (MRC-DTCRO). NSGA-II is proposed for solving this problem and it selects the best compromise of total project completion time, total project cost and resource fluctuations.

Initial population is created in the same way of Hartmann [65]. In other words, an activity precedence feasible list and a mode assignment list are used for representing the individuals. Activity list is constructed by choosing the activity from the decision set randomly. One-point crossover operator proposed by Hartmann [65] is applied. As for generation scheme, serial SGS is preferred by the authors.

Elazouni and Abido [48] solve the project scheduling problem under cash constraints. A SPEA with logic-preserving crossover and mutation operator is preferred by the authors. Profit values of the individual projects construct a set of conflicting objectives. The performance of the proposed algorithm is evaluated by comparing it with multi-objective GAs. Proposed algorithm provides the same solutions with those of GAs. Moreover, it is successful in obtaining well-diverged solutions.

Kılıç et al. [79] solve the project scheduling problem with each activity having risk. During defining risk of activities, various levels for risk and preventive operations for each level are defined. This problem has two objectives being the minimization of the expected Cmax and the expected total cost. Mixed integer programming model and GAs are proposed for solving this problem.

Additionally, improvement heuristic is proposed to reduce the total cost without changing the critical path and Cmax. This can be succeeded with a method resembling the trade-off case in multi-mode RCPSP.

Nikulin and Drexl [111] focus on airport flight gate scheduling problem with multi-objective, which is modeled as a multi-mode multi-criteria RCPSP. The objectives are the maximization of total flight gate preferences, the minimization of the number of towing activities and the absolute deviation of the new gate assignment from a so-called reference schedule. The solution method preferred is PSA. When there is uncertainty in input data, this problem is approached with fuzzy numbers.

Hanne and Nickel [60] deal with scheduling and inspection planning for software development projects. They propose multi-objective evolutionary algorithm and three

objectives should be minimized; the total number of defects, the  $C_{max}$  and the total cost. It is observed that this algorithm outperforms first come first serve simulation method in all of three objectives.

Xiong et al. [149] solve RCPSP with multi-objective. The activities have the stochastic durations and that is why, three different objectives are considered. They aim at minimizing  $C_{max}$ , maximizing robustness and stability. Hybrid multi-objective evolutionary algorithm incorporated with local search procedure is proposed. Specifically, NSGA-II is preferred as the solution method. As for the comparison of the algorithms, the performance measure called set cover is utilized.

Yannibelli and Amandi [151] consider project scheduling problem with multi-objective. While the minimization of  $C_{max}$  is aimed, assigning the most effective set of human resources to each activity is also aimed. The multi-objective hybrid search and the optimization algorithm is proposed and it is composed of multi-objective evolutionary algorithm and multi-objective simulated annealing. The integration of simulated annealing and evolutionary algorithm is performed in order to improve the search ability of the latter one. Parameters are chosen with preliminary experiments. Each instance is solved with different parameter combinations and the best combination resulting in the best and most stable results is chosen.

For evaluating the hybrid algorithm, some performance measures are defined. At first, the spread and distribution of the obtained set are analyzed. Moreover, accuracy of the related set is also analyzed. As for the comparison of the algorithms, several quality aspects of obtained non-dominated set are analyzed. Size and ratio of non-dominated solutions, accuracy, spread and distribution of the non-dominated solution sets are analyzed and a performance measure is proposed for each of these aspects.

### **2.2.3 Other formulations**

Vo $\beta$  and Witt [147] deal with the flow shop problem. They regard this problem as a RCPSP. Although they consider to minimize two objectives,  $C_{max}$  and weighted tardiness, they focus on the minimization of weighted tardiness because they see this problem from supply chain perspective. After assigning weights for both objectives, they set the weight value for  $C_{max}$  as 0. Thus, the problem turns into a single objective problem. In order to solve this problem, dispatching rules has been used by the authors. These rules are weighted earliest due date (WEDD), weighted latest finish time (WLFT) and weighted minimum slack (WMINSLK). However, these rules are modified to make them proper for

flow shop problem.

Bomsdorf and Derigs [13] focus on the Movie Shoot Scheduling problem, which can be seen a variant of RCPSP. However, the features existing in movie producing procedure make this problem much different than the classical RCPSP. Objectives considered in that study can vary. They are classified as the following: Minimization of location change, the minimization of  $C_{max}$  and satisfying the continuity during shooting the film, which means trying to shoot the movie in the order given by the movie script. However, quality of the schedules are evaluated by the following criteria: Changes of locations, length of gaps (it should be minimized), number of gaps, number of capacity renewals and continuity. Thus, the objective function is composed of all of these six criteria.

Cheng et al. [26] focus on job shop scheduling problem. They propose hybrid algorithm containing dispatching rules, the shifting bottleneck procedure and evolutionary algorithm. The objectives are the minimization of  $C_{max}$  and total tardiness. The authors make use of two performance indicators to compare the algorithms. These are binary hypervolume indicator developed by Zitzler and Thiele [154] and unary multiplicative epsilon indicator Zitzler et al. [155].

Esquivel et al. [50] deal with job shop scheduling problem with single objective and multi-objective. Evolutionary algorithm is used with its modified version. Multirecombination is included into this algorithm and a method is developed to prevent the premature convergence of the algorithm. In multirecombination approach, multiple crossovers per couple and multiple crossovers on multiple parents are applied.

Tamaki et al. [133] deal with identical parallel machine scheduling problem. The authors propose GA approach, where an individual represents the job orders in a machine and assignment of jobs to the machines. Starting time of the jobs are computed by solving the linear programming problem.

During solving the linear programming problem, each objective is assigned a weight value. The objectives are the minimization of the  $C_{max}$  and the minimization of earliness-tardiness costs. Thus, by changing the weight values, various schedules can be obtained.

Fowler et al. [55] deal with bi-criteria parallel machine scheduling problem. The main purpose of the authors is to develop a new performance measure to evaluate the algorithms and to make experiments for fine-tuning of the parameters because they believe that performance measure should be robust and efficient. For this purpose, Integrated Convex Preference (ICP) measure is developed and two different posteriori solution techniques based on GA are used for solving the problem. It is observed that ICP evaluates the approximation sets robustly, whereas other measures can misjudge the solution qualities.

The authors also classify all performance measures developed until that day. Afterwards, following decision maker's utility function, they develop their own performance measure called Integrated Convex Preference.

Jaskowski and Sobotka [74] pay attention to the construction project planning, scheduling and contractors selection. There are two objectives; Cmax and cost minimization. This problem is brought down to single objective form with the help of modified Tchebycheff achievement scalarizing function. Evolutionary algorithm is applied to this problem. As further research, the authors explain that parameter selection should be done with an experiment because they do not make use of such a method while setting the parameters.

Jaszkiewicz [75] develop a new genetic local search algorithm for solving multi-objective combinatorial optimization problem. At each iteration, the algorithm draws at random utility function, a couple of best individuals among the previous generation are selected and temporary population is constructed. Afterwards, a pair existing in this population is selected randomly and applied recombination. Local search procedure is applied to each of offspring. Computational experiments are performed on traveling salesman problem.

The author states that most of the performance measure depend on the fact that true Pareto front is known. However, this is not the case in that study. Thus, the quality of the approximate set is defined by the expected value of weighted Tchebycheff utility function over the set of normalized weight vectors.

Mansouri [100] deals with Just-In-Time sequencing problem with variation of production rates and number of setups' optimization, which are desired to be minimized. The author applies MOGA to this problem. This algorithm is compared to total enumeration and three heuristic solution methods. The results of the algorithms are evaluated by considering quality and diversity aspects. Fine-tuning of the algorithms are performed again with both aspects.

Osman et al. [114] think that while solving multi-objective dynamic programming problem, classical approaches convert to this problem into single objective case by assigning weights. This method requires to determine various weight values for each objective. Additionally, solving this problem becomes hard when the problem size increases. Thus, the authors propose GA for finding solution to this problem, specifically for multi-objective resource allocation problem.

## 2.3 Multi-project

### 2.3.1 Exact solution methods

Kramer and Hwang [90] develop a generalized resource constrained project scheduling model. The developed model can deal with any of the important scheduling objectives, alternative activity completion modes and scheduling multiple project simultaneously. The authors state that the model can be solved by mixed integer / linear programming software. The main issue of the developed model is to split the activities and to allocate different resource usage rates. At the end, numerical examples are presented and it is solved by a software.

Tiwari et al. [134] consider the situation in multiple project environment, where workers have different skills. It is said that multiple project RCPSP models this situation by assigning different durations for the activities conducted by the workers having different skills. However, the authors explain it becomes often inadequate. The real project that represents the customer training division of a large telecommunication company is studied and the labor assignment problem is modeled and solved by implementing integer programming optimization procedures. The project environment has multiple project being independent from each other but using the same resources. The results of the proposed method show the bottleneck of the resources and the benefit of cross-training of the labors. In addition, it guides which labors should be cross-trained in order to gain the highest benefit.

Mohanty and Siddiq [103] say that project managers in real life have to deal with multiple project simultaneously. Each project can have different number of activities and each activity has to use a certain amount of scarce resources in different times. Project managers must evaluate the performance of the projects with multiple criteria, but the literature has not many studies dealing with this duty. Therefore, the authors develop an integer goal programming, which then is analyzed with a case having three projects and three resources. Moreover, the extension of the model to deal with cost-time trade-off in managing the projects is presented.

Pritsker et al. [118] develop zero-one linear programming formulation for multiple project job-shop scheduling problems. They claim that this formulation is more general when compared to previously developed models. The formulation takes into account multiple resource constraints, due dates, job-splitting and the other real-world situations. Three objectives are tried, which are minimization of total time spent for finishing all

projects, minimization total throughput time for all projects and minimization of total lateness.

Krger and Scholl [91] explain that most of the researches assume resource transfers between projects without any cost and time. They think that this is not realistic and develop a new multiple project scheduling problem with transfer times and cost. At first, a framework is developed for involving this aspect to the problem, which considers managerial approaches to the transfer, types of transfer and the new role of the transferred resources. The problem is formulated as an integer linear programming problem. The experimental studies reveal that consideration of transfer of the resources is necessary and significant.

Chen and Askin [23] deal with project selection and scheduling problem because in real life, many project managers have to face such problems. The project return is defined as a function of project completion time and the objective is to maximize the present worth of profit. The resources considered in the study are renewable and limited. After presenting mathematical formulations for both this problem and its extensions, implicit enumeration algorithm that consists of project ordering and a prioritization rule for resource allocation is developed. The proposed algorithm has an embedded module for solving RCPSP at each stage.

Heimerl and Kolisch [68] schedule multiple IT projects by assigning multi-skilled workers with different efficiencies to the projects. The objective is to minimize the labor cost and the problem is formulated as mixed integer linear program with a tight LP-bound. The model is evaluated with regard to solution gap and computation time. The authors explain that applying mixed integer linear program is more beneficial than applying heuristics because the former one can obtain lower-cost and feasible solutions. Moreover, it is said that if central planning is adopted, then applying mixed integer linear program is more beneficial.

### **2.3.2 Genetic algorithms**

Ozmehmet Tasan and Gen [116] consider the network optimization problem. The conventional integrated selection and scheduling solution methods becomes complex, if the size of the problem gets bigger. Traditional methods aim at selecting the projects at first, then scheduling them. However, it is stated by the authors that these tasks have to be considered together because considering separately causes to loss of integrity. The authors propose the integrated genetic algorithm using the multi-stage decision approach.



In this study, two newly defined problems with different characteristics are solved with the proposed algorithm.

Goncalves et al. [57] deal with multi-project RCPSP. GA approach is proposed by the authors. Chromosome representation is achieved by using random keys. The schedules are formed with a heuristic, which builds the parameterized active schedules after the definition of release dates, delay times and priorities by GA. The proposed procedure is tested by using 10, 20, 30, 40 and 50 projects that have 1200, 2400, 3600, 4800, and 6000 activities, respectively. The results reveal that GA-SlackMod can provide better results when compared to the results obtained by two other approaches (GA-Basic and GA-SlackND).

Ramrez Palencia and Meja Delgadillo [119] deal with bus body assembly line in Colombia, South America. They implement a computer system for this assembly line. The sequencing of the assembly line is formulated as multiple project RCPSP. GA is proposed as the solution methodology because of the capability of it to handle the constraints of the production system. The algorithm is embedded into a software generating weekly schedule. Thus, this software leads to the significant improvements in the company such as increasing on-time delivery percentage from 65% to 85% and changing the organizational climate in the company.

Kumanan et al. [93] focus on scheduling multiple project with resource constraints in order to minimize overall Cmax. The author explain that the proposed methods such as mathematical techniques and heuristic algorithms for scheduling is appropriate for single project, but they become cumbersome and inefficient for multiple project scheduling. Thus, a GA with a heuristic approach is proposed for scheduling multi-project with resource constraints. The proposed method is tested with a numerical example and it is observed to be efficient.

### **2.3.3 Hybrid methods**

Chen and Shahandashti [24] deal with multiple project scheduling problem with multiple resource constraints. It is explained that several heuristic methods are proposed for solving this problem because the solution procedure is a complex and time-consuming task. However, each of heuristic algorithms are suitable for only one type of problem. Thus, the authors propose hybrid GA and SA procedure as the solution algorithms because both of these algorithms are generic and can be appropriate for all optimization problems. The validation of the proposed algorithm is illustrated with three real projects and three test

projects. The results show that the hybrid procedure has better performance than GA, SA and modified SA.

Kim et al. [80] develop a new hybrid genetic algorithm with fuzzy logic controller to solve multi-project RCPSP. It is argued that trying to solve this problem with the classical optimization techniques is not easy. The objectives are the minimization of total project time and total tardiness penalty. Fuzzy logic controller depends on the initialization of the revised serial method, which is better than non-preemptive scheduling. At the final phase of the study, it is observed that the proposed hybrid algorithm outperforms the conventional GAs and adaptive GA.

Chen [25] studies the RCPSP with multiple project for the maintenance of mineral-processing equipment at a copper mine in China. At first, he applies 0-1 goal programming to this problem. Because the problem size is large and the computing time is high, the author proposes a two-phase hybrid solution method. In phase 1, a feasible schedule is found with the help of a heuristic under resource constraints. While passing to the second phase, many decision variables and redundant constraints are eliminated from the model. In the second phase, implicit enumeration method is applied to the reduced problem. It is said that 90 percent of the variables and 96 percent of the constraints can be removed, thus this procedure is able to solve the medium size problem.

### **2.3.4 Priority Rules**

Browning and Yassine [17] claim that past researches have not revealed a clear guidance for the project managers about which activity priority rules should be preferred. Therefore, the authors deal with multiple project RCPSP and 20 different activity priority rules are tested on tremendous amount of instances. The objectives are the minimization of project lateness and portfolio lateness. One of the results is that some of commonly used and advocated priority rules show worse performance. Additionally, the authors state that project managers have to decide about which priority should be preferred by observing their local and global objectives.

Krüger and Scholl [92] explain that many studies in the literature have assumed that a resource can be transferred from one project to another without any cost in multiple project environment. In that study, sequence- or resource-dependent transfer times that can be regarded as setup activities are defined for the resources in order to complete this gap. The objectives are the minimization of mean project duration and priority rule based solution procedures are proposed. It is observed that if the scheduling scheme and priority

rule is chosen appropriately for each other, the procedures provide good results.

### **2.3.5 Auction mechanism**

Adhau et al. [2] say that dealing with multiple project is quite common and resource conflicts between project become a serious task for the project managers. In that study, two types of resources are considered. While one of them is always available to corresponding project, the other one is common for all projects. The existing multi-agent system using auction applies exact methods for solving winner determination problem so as to allocate the resources to the projects. However, this system is observed to fail to converge for some multiple project instances and unsuccessful for real life large projects. Thus, the author propose multi-unit combinatorial auction and a novel distributed multi-agent system using auctions based negotiation approach is used for solving winner determination problem.

Arazo et al. [6] focus on the project portfolio scheduling. Each incoming project affects the portfolio's schedule, resource availability and planned performance. It is stated that there is no analytical solution to schedule the resources dynamically. Mathematical approaches such as integer programming and network-based techniques can not be enough for such complex problems. In this paper, multi-agent system, where projects and resources are defined as agent and each project negotiate for the procurement of the resources through auction mechanism is proposed.

Confessore et al. [31] consider multiple project scheduling problem in which each project have several activities using special resources and common resources. The objective is the minimization of the duration of each project. The authors think that local decision makers representing projects should exist and combinatorial auction mechanism should be implemented. A dynamic programming formulation is proposed for auction mechanism and heuristic algorithms are proposed for both auction and bidding processes.

Homberger [70] try to solve the RCPSP with restart evolution strategy, which solves this problem through evolutionary algorithm repeatedly and uses the best solutions provided by each run. The instances having at most 2400 activities are solved with the provided method and the results show that it is better than the best-known heuristics. Integrating restart evolution strategy into multi-agent system, the authors solve decentralized resource constrained multiple project scheduling problem, in which each project has an agent who negotiates about the allocation of shared resources and schedules the activities using restart evolution strategy. To evaluate the proposed method, 80 generated

instances having up to 20 projects and up to 120 activities are solved. The results reveal that this method is competitive with central solution approach.

### **2.3.6 Other methods**

Hao et al. [61] focus on multiple project RCPSP. The authors firstly propose a dynamic project scheduling algorithm, which is based on partial task networks. It is stated that this algorithm can solve large scale RCPSP with complex time and resource constraints. For the project environment having multiple project, an interactive decision support system is involved to the algorithm and the resource conflicts are solved by this system. The proposed algorithm is applied and tested in a web-based aircraft maintenance management system.

Kao et al. [76] deal with uncertainty in project management. Specifically, during execution of multiple projects, some unexpected events occur and this leads to resource contentions among projects and schedule disruptions. With respect to this consideration, the authors adopt an event-driven approach in order to design a trade-off decision framework for project portfolio scheduling and rescheduling. High Level Petri nets, Activity-Based Costing and Technique for Order Preference by Similarity to Ideal Solution are performed in sequence to generate feasible schedules, estimate the Cmax and cost values and select the best compromise schedule.

Varma et al. [145] consider the pharmaceutical R&D projects and state that some critical decisions have to be made about project selection, activity scheduling and resource assignments to the projects. Pharmaceutical R&D activities have critical issues, which make it complex such as technological and market uncertainties, long development cycle times and work process constraints. The authors explain that in spite of these risks, the literature about allocating the resources among R&D activities is limited. A procedure called SIM-OPT is proposed by the authors to maximize the portfolios expected net present value, to control risk and to reduce drug development cycle times.

Coffin and Taylor III [30] say that the literature on R&D project selection focus on multiple criteria modeling of the problem. Mathematical programming models, decision theory models and scoring models are such models. However, these models do not take into account the project scheduling. The proposed model evaluates the single objective function, which represents the multiple criteria of the R&D selection and scheduling. The objectives are maximization of the expected profit for the portfolio, maximization of average probability of success for the portfolio, and minimization of Cmax of the portfolio.

The solution method includes fuzzy logic within a standard beam search approach. A real project is used for illustrating the procedure of the algorithm and benchmark studies are implemented to evaluate the efficiency of the algorithm.

Carazo et al. [21] consider the problem of project selection and scheduling. Multiple objectives conflicting often to each other, resource constraints, the transferring of the resources that are not used in the current period to the following period and project interdependence (complementarity, incompatibility, synergy and precedence relationships) are involved to the model. The author develop a multi-objective binary programming model, which is appropriate for both selection and scheduling objectives. The proposed solution method is a heuristic algorithm that is based on scatter search. In order to compare this method to other heuristics, instances are generated randomly and computational studies are performed. It is observed that the proposed procedure is stable against the changes of the characteristics of the problems. Additionally, it outperforms the SPEA2.

Sun and Ma [131] say that selection of R&D projects are studied, but scheduling them is considered rarely. The authors study the development and the application of packing-multiple-boxes model, which is a heuristic method and based on packing-single-box model. This model is said to be appropriate for both selecting and scheduling of R&D projects and it is used for both tasks of a case R&D company. The company conducts 20 projects simultaneously with financial constraints. Although the method may not be an optimistic one, it is said to be beneficial and efficient.

Lawrence and Morton [95] focus on RCPSP with multiple project and consider to minimize weighted tardiness costs. By extending their earlier proposed heuristic, they develop efficient and effective procedure so as to generate low cost schedules for the problem mentioned above. By considering the balance between the marginal cost of starting an eligible activity later and the marginal benefit of such an operation, a 'cost-benefit' scheduling policy is developed. This method is compared to a couple of dispatching rules taken from the literature and several new scheduling rules. The authors say that the results are in favor of 'cost-benefit' scheduling policy.

Gutjahr et al. [58] approach to Project Portfolio Selection problem with employees' multi-skills and evolutions with the objectives stated as economic gains and competency gains. The problem is divided into two sub-problems. Master problem is considered as project selection problem and subproblem is considered as slave problem dealing with assigning workforce to the selected projects. As for slave problem, asymptotic approximation of it by applying linearized formulation is proposed and efficient and exact solutions of the corresponding problem are found. For the master problem, NSGA-II and parallel

ant colony algorithm (P-ACO), which are multi-objective problem solving methods are compared. The randomly test instances and real world test instances are used for showing the results of computational analysis.

Kogan and Shtub [84] deal with multiple project RCPSP with variable-intensity activities to minimize the dynamic earliness and tardiness of project activities. Four dynamic models based on four types of precedence relations are studied. The first two models are created in order to deal with start-to-end relations with unit step function and specially constructed penalty functions, respectively. The last two models are designed to deal with overlapping precedence relations by considering the milestone approach and start-to-start precedence relations with lags, respectively. For solving the last three models, an efficient time-decomposition approach is adopted, which is used a guide for solving the first model.

Fatemi Ghomi and Ashjari [51] explain that resource constrained single project scheduling problem with stochastic task durations is a complex problem and this complexity becomes more when it has multiple project and common resources. The authors approach multiple project resource allocation problem (MPRA) as a multi-channel queuing system. The simulation modeling has been used to solve MPRA and the study of Fatemi Ghomi and Ashjari [51] develops a framework and a solution procedure by using simulation language GPSS. A numerical example is showed and the statistical results are presented.

Kim and Schniederjans [81] state that some real world scheduling problems are subjected to limited resources and this certainly turns out to be a challenge for the project managers. When it has multiple project, the complexity increases. The authors claim that conventional scheduling techniques such as PERT and CPM becomes insufficient to deal with this complexity due to their unrealistic assumptions. By making use of recent development in artificial intelligence and knowledge engineering with heuristic methods, expert systems are developed. The authors present a framework and development strategy for an expert system in multiple project scheduling domains. A practical application of the proposed system is presented as well.

Chiu and Tsai [27] deal with multiple project RCPSP with discounted cash flows by considering the project delay cost and early completion bonus. An efficient heuristic method is developed and proposed by the authors. To evaluate its performance, 42 small size instances are solved with it and the solutions of it are compared to optimal ones. It is observed that the solutions provided by the heuristic are very close to optimal. In addition, the heuristic algorithm is compared to existing four heuristics by solving single project and multiple project instances with the objectives of average total project net present value

and the average total project delay. The results show that the heuristic proposed by the author outperforms the other ones and the number of times that it obtains the best solution with those objectives are far more than that of other heuristics.

Lova et al. [99] explain that although many researches focus on scheduling single project and improves the objective function related to time, companies conduct several projects simultaneously and take into account no time objectives. Thus, they develop a multi-objective heuristic that improves one time type objective such as mean project delay or multiple project duration increase and one no time type objective such as project splitting, in-process inventory, resource leveling or idle resources. The proposed heuristic method consists of several algorithms based on the improvement of multiple project feasible schedules. After finding feasible schedules by priority rule based heuristics or project scheduling softwares, this method improves them by applying backward and forward passes.

Kurtulus and Davis [94] say that heuristic methods to solve the project scheduling problem are studied deeply, but there is not an exact guide for the project managers about which method should be preferred. In addition, since there is no categorization for the heuristic rules, it is generally assumed that once a rule is selected, it must be applied throughout the project. The authors develop a categorization for the heuristic rules based on two different performance measures, which are the location of the resource usage peak and rate of utilization of each resource type.

Kolisch [87] consider multiple, large-scale, make-to-order assembly scheduling, which consists of scheduling assembly operations belonging to different orders. In addition to classical project scheduling constraints such as precedence and resource constraints, spatial resource and part availability constraints are also taken into account. The objective is to minimize sum of weighted tardiness and the problem is formulated as mixed integer programming. The proposed solution method is a list-scheduling heuristic. To evaluate the effectiveness of it, some benchmark instances are solved with it.

Shtub et al. [126] consider the scheduling problem in the environment in which the same products are produced repeatedly. In this problem, the authors have to decide how many units of product should be assigned to an available individual or an available team. It is stated that these kind of problems are classified by two conflicting aspects: (i) the requirement to complete each product by its due date, (ii) learning effect. This problem is studied with two different penalty cost structures and the models are developed for both of them. Different heuristics, which are SA, GA and pair-wise swap heuristic, and exhaustive search are tested. The results make clear that pair-wise swap algorithm is the

best.



## Chapter 3

# A Multi-objective Genetic Algorithm Approach and Extensions

In this chapter, problem characteristics, especially the objectives and the objective combinations, are firstly defined. Then, the way of creating and modifying problem instances are explained in detail. As proposed in the literature, preprocessing of the instance files to reduce the solution space are described. For the main components of GA, the ways of forming the initial populations are explained firstly. The different ways of fitness calculation are presented in the next step. The various types of crossover and parent selection mechanisms are implemented in this thesis. The explanations of these mechanisms plus mutation and parent reduction mechanisms are presented. Since the developed algorithm is for solving the multi-objective RCPSP, some additional operators such as crowding distance and non-dominated sorting procedure are implemented, whose definitions are given at the last sections of this chapter. Finally, the basic schemes of GA are given.

RCPSP has been tried to be solved to optimality. Blazewicz et al. [10] states that this problem is  $\mathcal{NP}$ -hard. As stated before, many exact, heuristic and metaheuristic methods have been developed and proposed until so far. According to significant studies which review the heuristic and metaheuristic methods, GA is one of the successful metaheuristic algorithms to solve RCPSP. Hartmann and Kolisch [67] argue that GA and SA procedure are the most effective algorithms. Moreover, Kolisch and Hartmann [88] claim that GA is one of the most significant algorithms for solving this problem.

In this thesis, multi-mode multi-objective form of RCPSP is tried to be solved efficiently. As for solution procedure, GA is preferred due to the its successes mentioned above. In the literature, several algorithms have been proposed for solving the multi-

objective RCPSP. One of the most successful procedures is NSGA-II (Zitzler et al. [153]), which compares the individuals fast and maintains the diversity well (see Deb [39]).

There exist different objectives that are considered together in this thesis. At first, a list of the objectives are given below and the definitions of some of them are explained shortly, if required. The abbreviations inside the parenthesis will be used for these objectives.

- Minimization of Cmax: Minimization of the completion time of the last activity (minCMAX)
- Maximization of net present value (Npv): Maximization of total Npv computed by summing Npv of the costs and payments of the activities (maxNPV)
- Maximization of minimum outflow: Maximizing the minimum summation of the Npv of the costs (maxMO)
- Minimization of resource usage deviation (minRUD): For the definition of this objective, see the Equation (3.14) in Section 3.4
- Minimization of mean weighted tardiness of the projects (minMWT)
- Minimization of mean flow time of the projects (minMFT)
- Minimization of mean completion time of the projects (minMCT)

The last three objectives make sense, if the problem environment has multi-project. In implementation phase of this thesis, RCPSP is studied with both single project and multi-project. Thus, all of above mentioned objectives are meaningful for this thesis. The last three objectives are particularly relevant for multi-project. In other words, weighted tardiness values are calculated for the projects which have some certain due dates. In addition, average flow and completion time are also for the projects.

In this thesis, the objectives considered together while solving the multi-objective RCPSP are listed as below. The abbreviations next to the explanations will be used for the corresponding objective combinations.

- Min. of Cmax and Max. of Npv (minCMAX/maxNPV)
- Min. of Cmax & Max. of Npv & Min. of resource usage deviation (minC-MAX/maxMNPV/minRUD)
- Min. of Cmax & Max. of Npv & Max. of minimum outflow (minCMAX/maxMNPV/maxMO)

- Max. of Npv & Min. of mean weighted tardiness (maxNPV/minMWT)
- Max. of Npv & Min. of mean flow time of the projects (maxNPV/minMFT)
- Max. of Npv & Min. of mean completion time of the projects (maxNPV/minMCT)

Besides its ability to solve the multi-objective case, the algorithm implemented in our study can also solve the single objective case of RCPSP. These objectives are the minCMAX and the maxNPV.

## 3.1 Project instance

### 3.1.1 Modifications in single project instances

For implementing the algorithm and solving multi-objective RCPSP, some researchers have created data instances. Demeulemeester et al. [41] develop a library consisting of project scheduling networks and those network are updated by the study of Vanhoucke et al. [142]. Moreover, Kolisch and Sprecher [89] create another instance library for project scheduling problem. In this thesis, the library PSPLIB (project scheduling problem library) created by Kolisch and Sprecher [89] is preferred.

In order to solve RCPSP with the objective minCMAX, the original form of the instance files are used. In other words, no change or modification is operated. Each activity including dummy source and dummy sink activities has its own number of modes, the number of successors, the set of successor activities. Additionally, each mode of an activity has duration, resource usage amount for two renewable resources and two non-renewable resources. There exist a capacity for each resource.

On the other hand, since these files are not generated for the objective maxNPV or other monetary objectives, they do not include the corresponding information both for the project and the activities. That is, some changes are required to turn them into the appropriate form.

Firstly, for non-dummy activities, cash outflows are generated for each mode of each activity. For doing this, the sum of all resource usage amount  $sum_{jm}$  is obtained by the following equation:

$$sum_{jm} = \sum_{k=1}^K r_{jkm} + \sum_{i=1}^l w_{jim} \quad \forall j \in [1, N] \text{ and } \forall m \in [1, M_j] \quad (3.1)$$

Afterwards,  $raw_{jm}$  is calculated by the equation  $raw_{jm} = -10 \cdot sum_{jm}$ . It should be noted that if  $raw_{jm}$  is not decreasing with the increasing duration by considering all modes of the activity  $j$ , the order of the costs is modified and converted to the proper form. Following this stage, the largest cost is multiplied by 1.8, the second largest cost is multiplied by 1.2 and the least cost is multiplied by 1 (there are three modes for each activity excluding dummy source and sink activities). At the end of this procedure, the difference between costs are well distinguished and it allows the algorithm to select the most proper mode during maximizing Npv or when Npv is among the objectives during solving multi-objective RCPSP.

As for the dummy source and sink activities, the former one is not assigned any cost. However, the dummy sink activity is assigned a payment (cash inflow) called lump sum payment by applying the following procedure: The averages of the costs of the modes  $\bar{c}_j$  computed by the following equation

$$\bar{c}_j = \frac{\sum_{m=1}^{M_j} c_{jm}}{M_j} \quad \forall j \in [1, N] \quad (3.2)$$

are summed and the resulting value is multiplied by 1.1.

$$c_N = \sum_{j=1}^{N-1} \bar{c}_j \cdot 1.1 \quad (3.3)$$

At the end of these operations, the lump sum payment of the dummy sink activity denoted by  $c_N$  is assigned, where 1.1 represents the factor such that the payment of this activity can become large enough. This factor is determined intuitively for this purpose. It should be stated that it can take other values such as 1.2 and 1.3, if needed.

The payment amount of the dummy sink activity is assigned relatively large because the algorithm would increase the Cmax infinitely, if this value is not that large while maximizing the Npv. After these all modifications, the instance files are turned into the proper form being solved by this algorithm.

This operation is performed for only single project instance file. With these instances, fine-tuning of the parameters (will be explained later) are performed. The reason why this experiment is conducted with only single project instance files is to spend less time for it. In addition, multi-project instance files are created in this thesis. During creation, a different cost assignment technique is utilized. Therefore, there exist two different cost assignment techniques. It should be noted that the result of fine-tuning experiment con-

ducted with single project instance files are used for solving multi-project instance files. Another fine-tuning experiment whose scope is kept limited is conducted with multi-project instance files. The creation method of multi-project instance files are explained in the following:

### **3.1.2 Multi-project instance generation**

When GA runs with multi-project instances, two different instance sets are utilized. First of them is created by the study of Can and Ulusoy [20]. In these files, the names of combined single projects taken from PSPLIB are recorded. Each project is assigned a lump sum payment and investment value. While lump sum payment is realized at the end of the project, the investment is realized at the start of the project. In addition, the capacities for renewable and non-renewable resources are determined for major project. Finally, interest rate is recorded in the instance files.

Second multi-project instance set is created in this thesis. In addition to project features mentioned above, the due dates and weights of the projects are assigned. However, the investment is not assigned for the projects.

As stated before, the instances existing in PSPLIB do not have monetary objective. Thus, each activity excluding dummy sink activities of the projects should be assigned a negative cash flow. In both multi-project instance sets (created in Can and Ulusoy [20] and in this thesis), cost assignment technique proposed by Can and Ulusoy [20] is preferred, which will be explained later.

The main purpose of generating a new multi-project instance set is to obtain instances which have different difficulty with respect to due dates, lump sum payments and resource capacities. Thus, with these instances, GA is tested and its behavior is observed under different instances. It should be noted that the observation becomes reasonable if the values of one of the mentioned factors are changed and the values of the remaining factors are kept steady. For instance, if it is desired to test GA with varying due dates, lump sum payments and resource capacities should be kept steady.

Single projects having the same number of activities taken from PSPLIB are merged for generating multi-project instances. Additionally, single projects having different number of activities are also combined in order to observe how GA behaves under different size projects. The way of multi-project instance generation for the first case is defined in Table 3.1. In this table, the number of activities that multi-project has are shown.

Single 10 Projects	# of Activities for Multiple Projects	Single 15 Projects	# of Activities for Multiple Projects	Single 20 Projects	# of Activities for Multiple Projects
10 Activities	100	10 Activities	150	10 Activities	200
12 Activities	120	12 Activities	180	12 Activities	240
14 Activities	140	14 Activities	200	14 Activities	280
16 Activities	160	16 Activities	240	16 Activities	320
18 Activities	180	18 Activities	270	18 Activities	360
20 Activities	200	20 Activities	300	20 Activities	400
30 Activities	300	30 Activities	450	30 Activities	600

Table 3.1: Merging the Single Projects

### 3.1.2.1 Data generation for testing due date

Multi-project instances are generated for different due dates because some of the objectives require the projects to have due dates. For this purpose, the instances should have different due date difficulties. Due date difficulty implies how strictly due dates are set for the projects. Difficulty starts with 1, which denotes the most challenging situation and ends with 5, which denotes the easiest situation.

It is obvious that other factors, which are renewable and non-renewable resource capacities and lump sum payments should be determined clearly. It is considered that it might be difficult to assign rational values without solving these instances with GA or making some experiments. Thus, in order to have proper values for these factors, multi-project instances are solved and some insights about resource usages, Npv values and completion time of the projects are provided. The procedure is as follows:

Single projects presented in Table 3.1 are merged and a multi-project instance is provided. For doing so, the dummy source activity of each single project is deleted and a major dummy source and sink activity are inserted. Thus, each single projects has a predecessor activity, which becomes major dummy source activity and has a successor activity being dummy sink activity. In this case, this network represents a single project because there is no difference between single project and multi-project in terms of network structure (see Figure 3.1).

After constructing the project network, an individual that has precedence feasible activity list and mode assignment list are formed. Critical path method (CPM) is applied on this individual in order to assign the earliest finishing and the latest finishing times for each activity. It should be noted that the resource capacities that have not been already assigned are not taken into account during this procedure. With the help of CPM, the earliest finishing time of each projects are recorded.

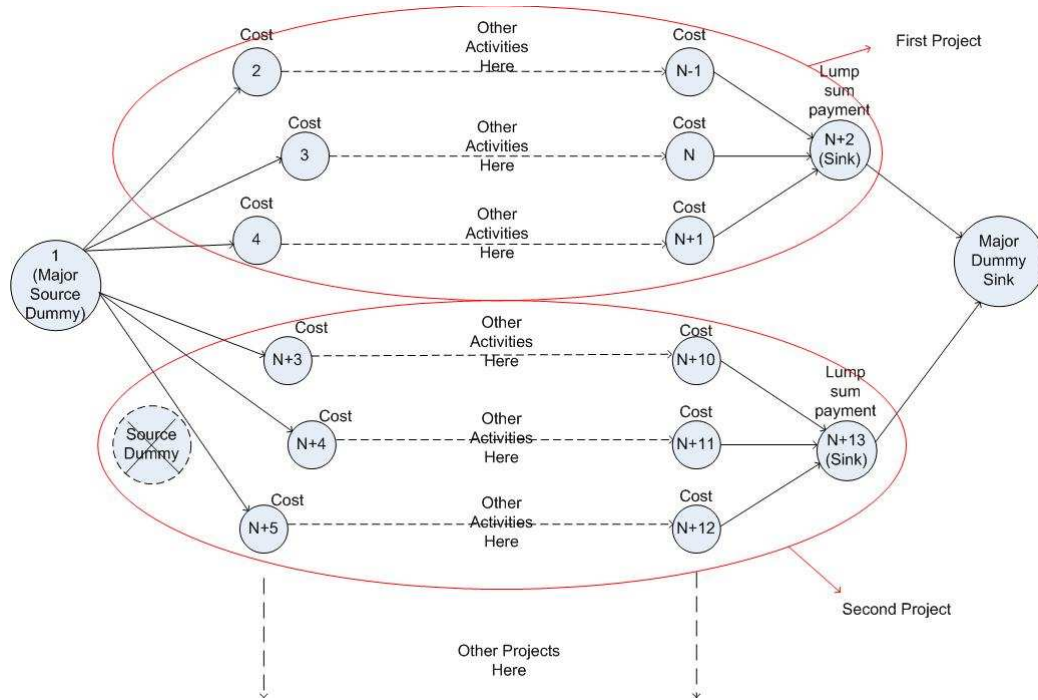


Figure 3.1: Multi-Project Network Structure

The procedure mentioned above is repeated by the number of activities of multi-project instance. Thus,  $EF_i^j$  that denotes the earliest finishing time of the project  $j$  in iteration  $i$  are recorded. The Table 3.2 summarizes this procedure. In this table  $A$  denotes the number of projects to be combined and  $B$  denotes the total number of activities existing in multi-project network.

As for renewable resource, the average usage of the resource over time horizon is computed and recorded for each iteration. Let  $r_{it}$  denote the renewable resource usage level in time  $t$  for iteration  $i$ . Then the Equation (3.4) gives the average value. Applying

Iteration	Projects			
	1	2	·	A
1	$EF_1^1$	$EF_1^2$	·	$EF_1^A$
2	$EF_2^1$	$EF_2^2$	·	$EF_2^A$
3	$EF_3^1$	$EF_3^2$	·	$EF_3^A$
·	·	·	·	·
·	·	·	·	·
·	·	·	·	·
B	$EF_B^1$	$EF_B^2$	·	$EF_B^A$

Table 3.2: The Earliest Finishing Times of The Projects

Iteration	Average Usage
1	$\bar{r}_1$
2	$\bar{r}_2$
3	$\bar{r}_3$
.	.
.	.
.	.
B	$\bar{r}_B$

Table 3.3: Average Usage of Renewable Resource

Iteration	Average Usage
1	$\bar{n}_1$
2	$\bar{n}_2$
3	$\bar{n}_3$
.	.
.	.
.	.
B	$\bar{n}_B$

Table 3.4: Average Usage of Nonrenewable Resource

the same process for each iteration, the Table 3.3 is constructed.

$$\bar{r}_i = \frac{\sum_{t=1}^{Cmax} r_{it}}{Cmax} \quad (3.4)$$

The average of non-renewable resource usage ( $\bar{n}_i$ ) is computed as in Table 3.4.

Npv values of each project should be calculated and observed to determine the lump sum payments properly. The cost of each activity is assigned as in the study Can and Ulusoy [20]. The following definitions are given to explain the procedure for cost assignment:

- $c_{jm}$ : cost of the activity  $j$  operating in mode  $m$
- $d_{jm}$ : duration of the activity  $j$  operating in mode  $m$ .
- $r_{jkm}$ : the amount of renewable resource  $k$  required to operate the activity  $j$  in mode  $m$ .
- $w_{jim}$ : the amount of non-renewable resource  $i$  required to operate the activity  $j$  in mode  $m$ .



Iteration	Projects				
	1	2	·	·	A
1	$Npv_1^1$	$Npv_1^2$	·	·	$Npv_1^A$
2	$Npv_2^1$	$Npv_2^2$	·	·	$Npv_2^A$
3	$Npv_3^1$	$Npv_3^2$	·	·	$Npv_3^A$
·	·	·	·	·	·
·	·	·	·	·	·
·	·	·	·	·	·
·	·	·	·	·	·
B	$Npv_B^1$	$Npv_B^2$	·	·	$Npv_B^A$

Table 3.5: The Npv of The Projects

- $\alpha_k$ : unit resource usage cost of utilizing one unit of renewable resource  $k$  for one period
- $\beta_i$ : resource usage cost of consuming one unit of non-renewable resource  $i$
- $K$ : the number of renewable resources
- $l$ : the number of non-renewable resources

$$c_{jm} = \sum_{r \in [1, K]} d_{jrm} \cdot r_{jkm} \cdot \alpha_k + \sum_{i \in [1, l]} w_{jim} \cdot \beta_i \quad (3.5)$$

In this thesis, the values  $\alpha_k$  and  $\beta_i$  are constant and equal to three. They are assumed to be equal to three in the study of Can and Ulusoy [20], as well.

With Equation (3.5), every activity except for dummy source and sink activities are assigned cost. Therefore, during calculation of Npv values of the projects, only the activities that are assigned costs are taken into account. The Table 3.5 summarizes this process ( $Npv_i^j$  denotes the net present value of the project  $j$  in iteration  $i$ ).

After completing the simulation, renewable and non-renewable resource value are assigned to instances. However, these should not change between instances because these instances are created for testing different due dates. Thus, the average value of  $\bar{r}_i$  and the average value of  $\bar{n}_i$  are assigned as renewable and non-renewable resource capacities, respectively. As for lump sum payments, the Table 3.5 is modified by the Equation (3.6).

$$Npv_i^j = \frac{-Npv_i^j}{(1 + e)^{EF_i^j}} \quad (3.6)$$

where  $e$  denotes the interest rate. After this procedure, the average of  $Npv_i^j$  for each project  $j$  is computed and assigned to the instance as lump sum payment of the project  $j$ .

In order to assign different due dates, the following procedure is developed:

Let  $DS$  represent the difficulty coefficient of due date, which takes the values in [0.8, 0.9, 1.0, 1.1, 1.2]. For each  $DS$  value, the Table 3.2 is modified by the Equation (3.7)

$$E_{ij} = E_{ij} \cdot \left( \log\left(\frac{B}{B_j}\right) + DS \right), \quad \forall i, j \quad (3.7)$$

where  $B_j$  denotes the number of activities of the project  $j$ . The function  $\log(\frac{B}{B_j})$  in this equation calculates the coefficient for determining how late due date should be for the corresponding project  $j$ . While calculating this coefficient, it takes into account the proportion of the total number of activities to the number of activities of the project  $j$ . For example, if total number of activities is 200 and the number of activities of the project  $j$  is 20, then this coefficient becomes 1. Otherwise, if project  $j$  has 10 activities, this means that due date of this project should be determined later than the previous one and the result of the calculation turns out to be 1.30103. In addition, with the help of  $DS$  values, the different due dates for a project  $j$  can be assigned. It is clear that the most difficult (the earliest) due date of the project  $j$  is calculated if  $DS = 0.8$  and the easiest (the latest) due date is calculated if  $DS = 1.2$ .

After modifying all of  $E_{ij}$  values with a  $DS$  value, the average value of  $E_{ij}$  for each project is assigned for the corresponding project  $j$  and for the current  $DS$ . Thus, if this process is repeated for all  $DS$  values, five distinct instances, where due dates are different can be created.

### 3.1.2.2 Data generation for testing lump sum payments

To generate the instances with resource capacities which have different difficulty, the same procedure with that of due dates is repeated. However, in the current case, due dates and lump sum payments should not change between the instances. During the simulation, Table 3.2 is formed again and the Equation (3.7) is calculated by using only  $DS = 1$ . The resulting values are determined as the due dates of the projects. In addition, Tables 3.3 and 3.4 are constructed with the same way and the same procedure is repeated for assigning resource capacities with the Equation (3.4). As for lump sum payments, Table 3.5 is again provided with the same way. However, Equation (3.6) is calculated with different difficulty coefficients, which have 0.8, 0.9, 1, 1.1 and 1.2 and the resulting values are assigned as lump sum payments of the projects.

### 3.1.2.3 Data generation for testing resource limits

The way of generating multi-project instances for testing resource capacity is the same with that of due dates and lump sum payments. However, due dates and lump sum payments should be the same between instances and calculated with the corresponding equations. As for resource capacities, Tables 3.3 and 3.4 are formed with the same way. While calculating the resource capacities for the instance, Equation (3.4) is calculated with different difficulty coefficients which have 0.8, 0.9, 1, 1.1 and 1.2. The resulting resource capacities are assigned for the instances.

### 3.1.2.4 Data generation with different number of activities

Beside the data generation with the same number of activities for testing due date, resource and lump sum payments difficulty, single projects having different number of activities should also be combined in order to test GA with again different due dates, resource capacities and lump sum payments.

For this purpose, the projects having 10, 20 and 30 activities are desired to be combined. In each multi-project instance, the number of 10-activity, 20-activity and 30-activity single projects can become either one, five or ten. For any combination of these numbers, the method mentioned in Section 3.1.2.1 for due date difficulty is applied. Thus, for each multi-project instance, 15 instances with different due date, resource capacity and lump sum payment difficulties are generated.

## 3.2 Preprocessing

Sprecher et al. [129] propose that some unnecessary modes and resources may be deleted from the project network without losing the feasible solutions and reducing the search space. The authors define three types of situations with respect to modes and resources.

Minimum and maximum request of the non-renewable resource  $i$  by the activity  $j$  is computed as follows, respectively:

$$\min(w_{ji}) = \min(w_{jim}, \forall m \in [1, M_j]), \quad (3.8)$$

$$\max(w_{ji}) = \max(w_{jim}, \forall m \in [1, M_j]), \quad (3.9)$$

After these definitions, the authors say that a mode is non-executable with respect to a

renewable resource  $k$ , if  $r_{jkm} > R_k, \exists j \in [1, N]$ . A mode is non-executable with respect to non-renewable resource  $i$ , if

$$\sum_{\substack{j=1 \\ j \neq z}}^N \min(w_{ji}) + w_{zim} > W_i. \quad (3.10)$$

A mode is said to be inefficient, if there exists another mode so that duration, renewable resource requirement and non-renewable resource requirement of the first one is larger than those of the second one. In addition, a non-renewable resource is called redundant, if the total of maximum usage of this resource for all activities is less than the capacity of this resource.

Preprocessing procedure is defined by the authors as follows: Firstly, all non-executable modes should be removed from the instance. Then, redundant non-renewable resources must be deleted. Afterwards, all inefficient modes need to be eliminated. At this point, if any modes are removed at the last step, second and third step must be repeated sequentially until any mode can not be removed in the last step. The reason why the second and third steps should be applied sequentially is that any operation (removing redundant non-renewable resources) in the second step might make some modes inefficient. Thus, they must be performed sequentially.

### 3.3 Forming the initial population

Evolutionary algorithms start the solution procedure by constructing the initial population. In the literature, different ways are employed to form the initial population. In this thesis, two different techniques are implemented for constructing the initial population: Random initial population generation and feasible initial population generation. The first method selects the activity for the next position randomly. However, since some instances are tough with respect to the capacities of the resources, initial population might be infeasible in this respect. Therefore, feasible initial population technique is proposed in our study.

Individual representation becomes a significant aspect of the initial population construction procedure. In this thesis, an individual  $I = (\lambda, \mu)$  is represented by a pair of precedence feasible activity list  $\lambda = (j_1, j_2, \dots, j_N)$  and mode assignment list  $\mu = (m_1, m_2, \dots, m_N)$ . That is to say, each activity in the precedence feasible list has a mode assigned from its mode set. This representation is defined by Ulusoy et al. [139] and by

Hartmann [65].

### 3.3.1 Precedence feasible activity list

Eligible activity is an activity whose predecessor activities have already been replaced in the feasible activity list. Precedence feasible activity list is constructed in two ways: In the first method, dummy source activity  $j_1$  is replaced into the first position of feasible activity list. Afterwards, one of the eligible activities is selected randomly and replaced for the next position. This operation is repeated until all activities are replaced in the feasible activity list. This procedure is called random sampling because each activity has an equal probability for being selected.

In the second method, regret-based biased random sampling of [?] is performed. In this method, each activity in the eligible activity set is assigned a selection probability, which is derived from relative portion of priority values calculated by the latest finish time of activities. For the details, interested readers are referred to Drexl [46] and [?].

In our study, among those two methods, random sampling is generally preferred.

### 3.3.2 Mode assignment list

Several mode assignment techniques are designed for this thesis. The reason for this variety is to try to construct feasible initial population with respect to non-renewable resources because each of those methods may be successful or unsuccessful for forming the feasible initial population.

#### 3.3.2.1 Random mode assignment

This method is described in the study of Hartmann [65]. The procedure is as follows:

Firstly, for each activity  $j$  in the precedence feasible list, a mode  $m_j$  is assigned from its mode set. Let the initial mode assignments for all activities become  $\mu_1$ . Then, the capacity over utilization  $L_{\mu_1}$  with respect to non-renewable resources is calculated. If  $L_{\mu_1} = 0$ , it means that feasible activity list with respect to non-renewable resources is obtained and mode assignment operation terminates. Otherwise, the second mode assignment  $\mu_2$  is performed. If  $L_{\mu_2} \leq L_{\mu_1}$ , then the current mode assignment is set to  $\mu_2$ . This operation is repeated until  $N$  unsuccessful trials (that is, it can not reduce  $L_{\mu_k}$ ) are performed or a feasible activity list with respect to non-renewable resources is obtained.

### 3.3.2.2 The mode with longest duration

This method is proposed in this thesis as a part of the effort to create a feasible initial population with regard to non-renewable resources. The procedure is defined as follows: For each activity  $j$  in the precedence feasible list, a mode having the longest duration comparing to other modes of the activity  $j$  is assigned.

### 3.3.2.3 The mode with minimum average utilization

This technique is also proposed in our study for the sake of getting feasible initial population. Let  $w_{jim}$  denote the requirement of the non-renewable resource  $i$  by the activity  $j$  operated in mode  $m$ .

In this method, for each activity  $j$  in the precedence feasible list, the mode  $m_j$  is assigned, where

$$m_j = (m_j \mid \min((\sum_{i=1}^l w_{jim})/l)), \quad \forall j \in [1, N] \quad (3.11)$$

### 3.3.2.4 Iterative assignment

Iterative assignment is another procedure proposed in our study for constructing the feasible initial population. Starting from the empty mode assignment list, the activities are reviewed iteratively. Let  $b_{it}$  denote the sum of the utilization of non-renewable resource  $i$  of all activities, which have been reviewed until the iteration  $t$ . In each iteration, the non-renewable resource  $i$  is selected, where

$$i = (i \mid \min(W_i - b_{it}), \quad \forall i \in [1, l]) \quad (3.12)$$

Afterwards, for determined non-renewable resource, the mode using this resource minimally comparing to other modes is selected for the corresponding activity. If other non-renewable resources are at the minimum value for this mode, then this mode is assigned for the corresponding activity. Otherwise, a mode is selected and assigned by reviewing the resource-resource trade-off between these resources.

Up until now, mode assignment techniques are defined. However, these techniques may be still unsuccessful to obtain a feasible individual with respect to non-renewable resources. Therefore, some repair procedures are proposed in this thesis. With the help of these repair procedures, an infeasible mode assignment might be changed to a feasible one.

### 3.3.2.5 Local repair

Given an infeasible mode assignment, this simple technique tries to convert it to a feasible one. This procedure visits the activities in the precedence feasible activity list starting from the first activity. Let  $t$  denote an iteration. For an iteration, this algorithm changes the mode of the activity. Then,  $b_{it}$  values, defined in Section 3.3.2.4 are updated. If  $b_{it} \leq W_i, \forall i \in [1, l]$ , then this procedure stops. Otherwise, it keeps the initial mode of the corresponding activity and continues visiting the next activity in the list. This algorithm terminates until a feasible mode assignment is provided or all activities are visited.

### 3.3.2.6 Extensive repair

Given an infeasible mode assignment, the sum of non-renewable resource usages by all activities are known. The non-renewable resource  $i$  whose sum is the most deviated one from the capacity  $W_i$  is determined and the activity operating this resource at the largest level comparing to other activities is selected. For this activity, all of its mode are reviewed and the mode whose determined non-renewable resource usage level is the minimum when compared to other modes is assigned to the corresponding activity. If the new mode assignment turns out to be feasible, then this procedure terminates. Otherwise, it continues repeating the same operations until a feasible mode assignment is obtained or all activities in the precedence feasible activity list are visited.

## 3.3.3 Random and feasible initial population

As stated before, random initial population means that feasible activity list in each individual of the population is constructed randomly. That is to say, an activity in eligible set is selected for the next position randomly.

Random initial population is proposed by Hartmann [65]. Let  $POP$  denote the desired population size, then the procedure is summarized as in the Algorithm 1.

As mentioned above, this thesis has the second method for providing the feasible initial population. Since the method of Hartmann [65] to convert an infeasible individual to a feasible one (see Section 3.3.2.1) is observed to be unsuccessful by making several experiments on different instances, we need to develop a method in order to find feasible initial population. First of all, in Phase I of this procedure, it tries to find a feasible individual with respect to non-renewable resources. In Phase II called as tree structure, it provides other feasible individuals by changing a mode of an activity on the feasible individual found at Phase I.

---

**Algorithm 1: Random Initial Population**

---

```
1 repeat
2   foreach Individual I do
3     Get activity list (see Section 3.3.1);
4     Assign modes (see Section 3.3.2.1);
5     Schedule the activities (to be explained later);
6   end
7 until The population size reach POP;
```

---

**\* Phase I**

Phase I uses all of the mode assignment techniques defined above. In addition, after each assignment method, obtained mode assignment is checked whether it is feasible. Moreover, previously defined repair functions are performed in certain places of Phase I. The detailed procedure is given in Algorithm 2.

After terminating the Phase I, it turns out to be that there are exactly two situations about the individual. If Phase I is able to get a feasible mode assignment, then feasible initial population procedure is continued with Phase II. Otherwise, feasible initial population procedure is canceled and random initial population procedure is performed. In this case, each individual is assigned a fitness value, which is much poor compared to standard fitness value.

**\* Phase II**

If Phase I is able to provide a feasible individual, then phase II of this algorithm tries to generate different feasible individuals. After finding a feasible individual, this algorithm keeps and labels it as the first level of tree structure. It should be noted that there does not exist another feasible individual in the first level.

For the second level of the structure, the algorithm selects the activity in the precedence feasible of the first level individual iteratively. It is checked whether the individual would again become feasible with another mode of the selected activity. If this is satisfied, then a new individual is created with the changed mode of the corresponding activity and this newly created individual is labeled as the second level of tree structure. If this does not satisfy, then the other mode of the selected activity is checked with regard to feasibility. Second level of tree structure is completed after reviewing all activities.

As for the following levels of the structure, the individuals created for the previous level are visited iteratively and the same operation is repeated for those individuals with only exception that the activities whose mode has been changed in previous levels can not be reviewed again. It should be stated that the number of branches to be created is set as



---

**Algorithm 2:** Phase I of Feasible Initial Population

---

```
1 Get activity list (see Section 3.3.1);
2 Assign modes as defined in The Mode with Longest Duration;
3 if Mode assignment is feasible then
4 |   Terminate the procedure;
5 else
6 |   Assign modes as defined in Random Mode Assignment;
7 |   if Mode assignment is feasible then
8 | |   Terminate the procedure;
9 |   else
10 | |   Assign modes as defined in The Mode with Minimum Average;
11 | |   if Mode assignment is feasible then
12 | | |   Terminate the procedure;
13 | | |   else
14 | | | |   Apply Local Repair;
15 | | | |   if Mode assignment is feasible then
16 | | | | |   Terminate the procedure;
17 | | | |   else
18 | | | | |   Assign modes as defined in Iterative Assignment;
19 | | | | |   if Mode assignment is feasible then
20 | | | | | |   Terminate the procedure;
21 | | | | | |   else
22 | | | | | | |   Apply Extensive Repair;
23 | | | | | | |   Terminate this procedure;
24 | | | | |   end
25 | | |   end
26 |   end
27 end
28 end
```

---

two.

Tree structure continues by applying the same operations until the number of individuals created equals to the *POP*.

### 3.4 Fitness calculation

After constructing the precedence feasible activity list and the mode assignment list, an individual must be assigned a fitness value. When it solves the single objective RCPSP, the fitness calculation is made by either considering Cmax or Npv value. On the other hand, if it deals with the multi-objective RCPSP, the fitness calculation is replaced with rank computation, which is achieved by classifying the individuals according to their domination over other individuals. An individual is said to dominate another individual, if the former one is better than the latter one with respect to the objectives. At the end of the rank computation, the rank value of an individual becomes one, if none of the individuals in the population dominates it. The detailed procedure can be observed in Section 3.6.

Scheduling means to assign starting and ending times for each activity existing in the feasible activity list. Basically, serial schedule generation scheme (SGS) is applied in order to obtain a scheduled feasible. Serial SGS schedules the next unscheduled activity at the first precedence and resource feasible time. Precedence feasible starting time is obtained by applying forward recursion with assigned modes. Serial SGS is preferred by many authors such as Hartmann [65].

Serial SGS generates always feasible schedules, which are optimal for RCPSP. Kolisch [86] has shown that serial SGS generates active schedules. Active schedules are defined that none of the activities can be started earlier without delaying some other activity. For scheduling problems with regular performance measures, there will always be an optimal solution in the set of active schedules. On the other hand, while parallel SGS generates feasible schedules for RCPSP as well, it constructs non-delay schedules. Non-delay schedule is a schedule that even if activity preemption is allowed, none of the activities can be started earlier without delaying some other activity. It is stated that the set of non-delay schedules is a subset of the set of active schedules. Therefore, it has smaller cardinality. Parallel SGS is said to have a disadvantage that it might not have an optimal solution for regular performance measures. Thus, serial SGS is preferred in this thesis.

It should be stated that another version of scheduling scheme is applied in this thesis. In Ulusoy et al. [139], a scheduler is proposed and in addition to two conditions, which

should be satisfied during serial SGS, a new condition should also be satisfied. Each activity can not be start earlier than the starting time of any activity existing at earlier locations in feasible activity list. For the detailed information, we refer to Ulusoy et al. [139].

After completing the starting and ending time assignment operation, the objective values of an individual can be calculated. Firstly,  $C_{max}$ , which is denoted by  $C_{max_I}$  is determined as the ending time of the dummy sink activity. The other objective  $Npv$  is calculated as below:

$$Npv_I = \sum_{j=1}^N c_j \cdot (1 + e)^{-SF_j} \quad (3.13)$$

where  $c_j$  denotes the cost of the activity  $j$  with assigned mode,  $e$  represents the interest rate and  $SF_j$  denotes the finishing time of the activity  $j$  after scheduling. Resource usage deviation of the individual ( $RUD_I$ ) can be computed by applying following equation:

Let  $r_{jk}$  denote the usage amount of the renewable resource  $k$  of the activity  $j$  with its assigned mode.

$$RUD_I = \frac{\sum_{j=1}^N \left| r_{jk} - \frac{\sum_{j=1}^N r_{jk}}{N} \right|}{N} \quad (3.14)$$

It should be stated that if the number of renewable resources is exactly one, the equation stays the same. However, if it is more than one, the average value over the number of renewable resources  $K$  of the above equation is assigned as  $RUD_I$ .

As for minimum outflow, time increment becomes an important aspect of this objective. It represents the minimum of sum of costs, which are realized throughout the project. In addition to the costs, payments are also taken into consideration during calculating this objective. For clarity, one can observe the Algorithm 3.

The algorithm 3 does not make sense in the case that single project RCPSP is being solved because every activity has already negative costs and the dummy sink activity has lump sum payment. Thus, it does not need to review the individual by incrementing the time because finishing time of dummy sink activity is  $C_{max}$ . On the other hand, this algorithm is especially necessary for the multi-project case because an activity having negative cash flow may finish later than dummy sink activities.

It should be noted here that some multi-project instances require to invest at the be-

---

**Algorithm 3:** Calculating the Minimum Outflow

---

```
1 Initialization;
2  $t$ : denotes the time;
3  $F$ : The set of activities finishing at time  $t$ ;
4  $MO_I$ : denotes the minimum outflow value of the individual  $I$ ;
5  $q$ : a float value;
6 for  $t = 0$  to  $Cmax$  do
7   Review all activities;
8   Construct the set  $F$ ;
9   foreach activity  $j \in F$  do
10     $q = q + \sum_{j \in F} c_j \cdot (1 + e)^{-SF_j}$ ;
11  end
12  if  $q < MO_I$  then
13     $MO_I = q$ ;
14  end
15 end
```

---

gining of the projects. Thus, the procedure 3 must be modified so that if a project starts at any time  $t$ , the investment must be realized in addition to the costs and the payments of the other activities.

Remaining objectives that are weighted tardiness, average flow time of the projects and average completion time of the projects are obviously for the multi-project case. By classical definition of the tardiness, weighted tardiness value of the individual can be easily calculated. Flow time of a project represents the difference between finishing and starting time of it. Thus, average value can also be calculated easily. The last objective for an individual is average completion time of the projects.

As stated above, the fitness calculation is performed, if only single objective case of RCPSP is being solved. When the objective is the minimization of  $Cmax$ , the fitness is assigned as  $Cmax$  value, if the individual becomes feasible with respect to non-renewable resources. Otherwise, the fitness value of an individual is assigned as

$$Fitness_I = Cmax_I + horizon; \quad (3.15)$$

where horizon represent the maximum value that  $Cmax$  can take. Thus, infeasibility of the individual is penalized by assigned a worse value. If the objective is the maximization of  $Npv$ , the fitness value is determined as  $Npv$  value of the individual, if it is infeasible. Otherwise, it is assigned the value selected as poor enough.

## 3.5 Forming the next generation

Some operators such as crossover, mutation, parent selection and population reduction must be performed for GA to proceed.

### 3.5.1 Crossover

In this study, three different crossover operants are preferred.

#### 3.5.1.1 One-point crossover

This crossover technique generates two individuals, which are called daughter and son from two parents called mother and father. In this type of crossover, two random integers  $q_1, q_2 \in [1, N]$ , where  $N$  denotes the number of activities in precedence feasible activity list are drawn. The activities existing in the positions from 1 to  $q_1$  in mother's feasible list are transferred to daughter feasible list. Afterwards, all activities in father's feasible list are visited and those, which are not replaced into daughter's list are transferred. As for modes, the activities holding the positions from 1 to  $q_2$  in daughter feasible list must have the same modes with those in mother's list. The remaining activities should take the same modes with those in father's list. For simplicity, the pseudocode of one-point crossover is given in Algorithm 4.

This type of crossover operant is defined by Hartmann [64] without considering the mode list and used by Hartmann [65] with redesigning it by considering the mode list.

#### 3.5.1.2 Two-point crossover

This crossover technique also creates two individuals from two parents.

Two random integers  $q_1, q_2 \in [1, N]$  are drawn. The activities holding the positions from 1 to  $q_1$  in mother's feasible list are carried to daughter's list. For the positions from  $q_1 + 1$  to  $q_2$  in daughter's list, all activities in father's list are reviewed and those, which do not exist in daughter's list are replaced until the position  $q_2$ . The remaining positions in daughter feasible list, activities in mother's list, which do not exist in daughter's list are transferred.

For assigning modes, two random integers  $q_3, q_4 \in [1, N]$  are drawn again. For the first  $q_3$  positions, the modes of the activities holding these position in mother's list are transferred to the corresponding activities. The modes of the activities between  $q_3$  and  $q_4$  are carried from the father's list. The remaining activities must hold the the same modes

---

**Algorithm 4: One-Point Crossover**

---

```
1 Initialization;
2  $p_c$ : Crossover rate;
3 foreach pair of mother and father list,  $M \& F$  do
4   Generate a decimal number  $p \in [0, 1]$ ;
5   if  $p < p_c$  then
6     Generate two random integers  $q_1, q_2 \in [1, N]$ ;
7     foreach position  $x \leq q_1$  in daughter's feasible list ( $D$ ) do
8       |  $D_x = M_x$ ;
9     end
10    foreach position  $x > q_1$  in  $D$  do
11      |  $D_x = F_x, z = \min(z \mid F_z \notin D)$ ;
12    end
13    foreach activity  $j \in D$  do
14      | if position of  $j \leq q_2$  then
15        |  $m_j = m_y, ID(j) = ID(y), y \in M$ ;
16      | else
17        |  $m_j = m_y, ID(j) = ID(y), y \in F$ ;
18      | end
19    end
20    Produce the son's feasible list with the same way, but by exchanging  $M$  and
     $F$ ;
21  else
22    Skip to the next pair;
23  end
24 end
```

---

of the mother feasible list. For simplicity, the pseudocode of two-point crossover is given in Algorithm 5.

---

**Algorithm 5:** Two-Point Crossover

---

```

1 Initialization;
2  $p_c$ : Crossover rate;
3 foreach pair of mother and father list,  $M \& F$  do
4   Generate a decimal number  $p \in [0, 1]$ ;
5   if  $p < p_c$  then
6     Generate two random integers satisfying  $q_1 < q_2 \in [1, N]$ ;
7     Generate two random integers satisfying  $q_3 < q_4 \in [1, N]$ ;
8     foreach position  $x \leq q_1$  in daughter's feasible list ( $D$ ) do
9       |  $D_x = M_x$ ;
10    end
11    foreach position  $q_1 < x \leq q_2$  in  $D$  do
12      |  $D_x = F_z, z = \min(z \mid F_z \notin D)$ ;
13    end
14    foreach position  $x > q_2$  in  $D$  do
15      |  $D_x = M_z, z = \min(z \mid M_z \notin D)$ ;
16    end
17    foreach activity  $j \in D$  do
18      | if position of  $j \leq q_3$  then
19        |  $m_j = m_y, ID(j) = ID(y), y \in M$ ;
20      | else if position of  $j > q_3$  and position of  $j \leq q_4$  then
21        |  $m_j = m_y, ID(j) = ID(y), y \in F$ ;
22      | else
23        |  $m_j = m_y, ID(j) = ID(y), y \in M$ ;
24      | end
25    end
26    Produce the son's feasible list with the same way, but by exchanging  $M$  and
27     $F$ ;
28  else
29    | Skip to the next pair;
30  end

```

---

This crossover technique is defined by Hartmann [64] for single mode project scheduling. Therefore, we create two more random integers for mode assignment procedure in this thesis.

### 3.5.1.3 Multi component uniform order-based crossover (MCUOBC)

This type of crossover operant, which is defined in the study of Ulusoy et al. [139], generates one individual called child from two parents.

For each iteration, a position in child's feasible list is replaced with an activity beginning from the first position. In an iteration, a random integer  $q$ , which can take the values either 0 or 1 is drawn. If  $q = 0$ , then the position in child's list is taken from the mother. Otherwise, the related activity of the father's list is transferred to the corresponding position.

After completing the activity list of the child, the mode list is subject to crossover. For each iteration starting from the first activity in child's list, this algorithm checks the modes of this activity existing in father and mother's list. If those modes are the same, then it does not need to do any operation for mode and that mode is transferred without changing. However, if those modes are different, a random integer  $q$ , which can take the values either 0 or 1 is drawn. If  $q = 0$ , the mode of that activity is taken from the mother. Otherwise, it is taken from the father.

The pseudocode of this kind of crossover operant is given in Algorithm 6.

## 3.5.2 Mutation

Each newly generated individual is applied mutation, which is defined in the study of Hartmann [65]. Both precedence feasible activity list and mode assignment list are subject to mutation operator.

For each iteration starting from the first activity in child's feasible list, that activity and the next activity are checked whether they have a predecessor relationship. If there is not any predecessor relationship between them and the mutation rate is satisfied, then those are exchanged. For the related iteration / position, if mutation rate is satisfied again, then the mode selected randomly is assigned for this activity. The pseudocode of this procedure is given in Algorithm 7.

## 3.5.3 Parent selection

In this thesis, two variants of parent selection operator are applied. Parent selection is applied on the population. According to the types of crossover, the number of pairs to be selected can vary. If one-point and two-point crossovers are applied, the number of pairs should be equal to  $\frac{POP}{2}$  because each pair generates two individuals. Thus, at the end



---

**Algorithm 6:** Multi Component Uniform Order-Based Crossover

---

```
1 Initialization;
2  $p_c$ : Crossover rate;
3 foreach pair of mother and father list,  $M \& F$  do
4   Generate a decimal number  $p \in [0, 1]$ ;
5   if  $p < p_c$  then
6     foreach position  $x$  in offspring's feasible list ( $O$ ) do
7       Generate a random integer  $q \in [0, 1]$ ;
8       if  $q = 0$  then
9          $O_x = M_x$ ;
10      else
11         $O_x = F_x$ ;
12      end
13    end
14    foreach activity  $j \in O$  do
15      if  $m_j \in M = m_j \in F$  then
16         $m_j \in O = m_j$ ;
17      else
18        Generate a random integer  $q \in [0, 1]$ ;
19        if  $q = 0$  then
20           $m_j \in O = m_j \in M$ ;
21        else
22           $m_j \in O = m_j \in F$ ;
23        end
24      end
25    end
26  else
27    Skip to the next pair;
28  end
29 end
```

---

---

**Algorithm 7: Mutation**

---

```
1  $P$ : The predecessor relationship;  
2  $p_m$ : Mutation rate;  
3 foreach position  $x$  in offspring feasible list  $O$  do  
4   | Generate a random decimal number  $q_1 \in [0, 1]$ ;  
5   | if  $q_1 < p_m$  and  $(O_x, O_{x+1}) \notin P$  then  
6   |   | Exchange the activities;  
7   | end  
8   | Generate a new random decimal number  $q_2 \in [0, 1]$ ;  
9   | if  $q_2 < p_m$  then  
10  |   | Assign a mode randomly for the activity  $O_x$ ;  
11  | end  
12 end
```

---

of crossover operators, the population size turns out to be  $2 \cdot POP$ . On the other hand, while applying MCUOBC, the number of pairs should be equal to  $POP$  since each pair generates one individual.

### 3.5.3.1 Roulette wheel selection

Roulette wheel selection basically depends on assigning selection probability varying with the fitness value or rank value. That is, the individual, which has better fitness or rank value is assigned larger selection probability compared to other individuals existing in the population.

Probability assignment procedure depends on whether single objective or multi-objective RCPSP is solved. Thus, if the objective is the minimization of  $C_{max}$  or the maximization of  $N_{pv}$ , then individuals in the population are assigned a probability value with regard to fitness value. Otherwise, the probability values are assigned depending on the rank values.

For the multi-objective case of RCPSP, this procedure works as in Algorithm 8.

In that case, the individual, which has the smallest rank value can have the largest probability of selection.

In the case that GA tries to minimize  $C_{max}$  as single objective, the procedure nearly stays the same, the only exception being fitness value of the individual is considered instead of rank value. On the other hand, if it maximizes the  $N_{pv}$ , then the algorithm sorts the population in decreasing order of fitness value and assigns selection probabilities by considering the fitness values instead of rank values.

---

**Algorithm 8:** Roulette Wheel Selection

---

```
1 Initialization;
2  $|Pair|$ : The number of pairs selected currently equal to 0;
3  $Prob_I$ : Selection probability of the individual  $I$ ;
4  $CumProb_I$ : Cumulative selection probability of the individual  $I$ ;
5  $Rank_I$ : Rank value of the individual  $I$ ;
6 Sort the population by increasing order of rank;
7 repeat
8   foreach Individual  $I$  in population do
9      $Prob_I = \frac{1 - \frac{Rank_I}{\sum_{I \in Population} Rank_I}}{POP-1}$ ;
10     $CumProb_I = \sum_{until I} Prob_I$ ;
11  end
12  Generate a random decimal number  $q \in [0, 1]$ ;
13  Find the interval  $CumProb_{I''} < q < CumProb_{I'}$ ;
14  Assign  $I'$  as mother;
15  Repeat this procedure without considering the currently assigned individual;
16  Assign the resulting individual as father;
17   $|Pair| = |Pair| + 1$ ;
18 until  $|Pair| = (POP/2)$ ;
```

---

### 3.5.3.2 Binary tournament selection

Binary tournament selection compares two individuals with respect to their fitness or rank values. The individual having the better fitness or rank value is accepted as the winner. Additionally, if those individuals have the same fitness values, then one of them is selected randomly. However, if the rank values are equal, then crowding distance operator takes the responsibility to determine the winner.

Binary tournament selection has a feature called crowding distance, which basically maintains the diversity of the evolutionary process. The detail procedure of the crowding distance operator will be explained in Algorithm 12.

The pseudocode of the binary selection operator is given in Algorithm 9.

---

**Algorithm 9:** Binary Tournament Selection

---

```
1 Initialization;
2  $Distance_I$ : The crowding distance value of the individual  $I$ ;
3 repeat
4   Select two individuals ( $I'$ ,  $I''$ ) from the population randomly;
5   if  $Rank_{I'} < Rank_{I''}$  then
6     Assign  $I'$  as mother;
7   else if  $Rank_{I'} > Rank_{I''}$  then
8     Assign  $I''$  as mother;
9   else
10    if  $Distance_{I'} > Distance_{I''}$  then
11      Assign  $I'$  as mother;
12    else
13      Assign  $I''$  as mother;
14    end
15  end
16  Select two individuals ( $I'''$ ,  $I'''' \neq mother$ ) from the population randomly;
17  Do the same operations for assigning a father;
18   $|Pair| = |Pair| + 1$ ;
19 until  $|Pair| = (POP/2)$ ;
```

---

### 3.5.4 Population reduction

After applying the crossover, the population size becomes  $2 \cdot POP$  with parent and offspring individuals. Therefore, this size must be decreased to again  $POP$ . For this procedure called population reduction, several techniques in the literature have been developed for both single objective and multi-objective case of RCPSP.

In this thesis, reduction operator has also been implemented with various forms for single objective RCPSP. These are the best ones, random reduction and elitist reduction.

#### **3.5.4.1 The best ones**

In this type of reduction, the best individuals with respect to fitness value are kept and the remaining ones are deleted from the population.

#### **3.5.4.2 Random reduction**

The individual that is randomly selected is deleted from the population. This procedure is repeated until the number of individuals existing in the population is  $POP$ .

#### **3.5.4.3 Elitist reduction**

In this type of reduction, a couple of the best individuals are left in the population and the remaining ones that will continue existing in the population are selected randomly.

It is obviously a strategy to determine the number of the best individuals to be left in the population. In our study, this number is accepted as the half of the population size.

As for multi-objective case of RCPSP, reduction procedure is implemented as described in the study of Deb [39]. According to this procedure, the best individuals are selected starting from the individuals whose rank value is equal to one. The pseudocode of the procedure is given in Algorithm 10.

## **3.6 Non-dominated sorting procedure**

As stated before, fitness calculation method is designed for only single objective. For multi-objective, this calculation is performed with fast non-dominated sorting designed by Deb [39]. During this procedure, each individual is labeled with a rank value according to domination principle.

Domination principle states that individual  $I'$  dominates the individual  $I''$  ( $I' \succ I''$ ), if  $I'$  is strictly better than  $I''$  for at least one objective and  $I'$  is better than or equal to  $I''$  for the remaining objectives. The pseudocode of this process is given in Algorithm 11.

In this algorithm, an individual is said to be rank  $z$  if  $i \in Front_z$ . By applying this procedure, a list of individuals are sorted in a fast manner. Moreover, several fronts, which represents the rank values of individuals are constructed.

---

**Algorithm 10: Population Reduction**

---

```
1 Initialization;
2 Next: The set of individuals that will continue for the next generation;
3 Frontq: The set of individuals whose rank value is q;
4  $q = 1$ ;
5 Frontqp: The individual in the  $p^{th}$  position of Frontq;
6 repeat
7   foreach Individual  $I \in \textit{Front}_q$  do
8     |  $\textit{Next} = \textit{Next} \cup I$ ;
9   end
10  |  $q = q + 1$ ;
11 until  $|\textit{Next}| + \textit{Front}_q > POP$ ;
12 Calculate crowding distance values  $\forall I \in \textit{Front}_q$ ;
13 Sort Frontq in decreasing order of distance value;
14 repeat
15   |  $\textit{Next} = \textit{Next} \cup I, (I | I \in \textit{Front}_q^1)$ ;
16   | Delete the individual  $I \in \textit{Front}_q^1$ ;
17 until  $|\textit{Next}| = POP$ ;
```

---

### 3.7 Crowding distance

Divergence is one of the most significant components of GA. In non-dominated sorting procedure, divergence is satisfied with crowding distance operator defined by Deb [39]. Each individual is assigned a distance value relative to other individuals existing in the same frontier. In other words, crowding distance value of an individual represents how it is close in terms of objective space to other individuals existing in the same front. This operator is used by binary tournament selection and population reduction. In both of them, crowding distance takes the responsibility to determine which individual becomes the winner in the case that rank values have equality. The detailed procedure is given below:

Let  $OB$  become the set of objectives. That is, it includes all of the objectives that are being together such as minimization of  $C_{max}$  and maximization of  $N_{pv}$ . Let  $OB^f$  be the  $f^{th}$  objective in this set and  $OB_h^f$  denote the objective value of  $h^{th}$  individual for this objective. Thus, the detailed procedure can be seen in Algorithm 12.

---

**Algorithm 11: Nondominated Sorting**

---

```
1  $SP_I$ : The set of dominated individuals by the individual  $I$ ;  
2  $N_I$ : The number of individuals dominating  $I$ ;  
3  $z = 1$ ;  
4 foreach Pair of  $I', I'' \in \text{population}$  do  
5   | if  $I' \succ I''$  then  
6   |   |  $SP_{I'} = SP_{I'} \cup I''$ ;  
7   | else  
8   |   |  $N_{I'} = N_{I'} + 1$ ;  
9   | end  
10 end  
11  $Front_z = Front_z \cup I, \forall I \mid N_I = 0$ ;  
12 repeat  
13   | foreach  $I' \in Front_z$  do  
14   |   | foreach  $I'' \in SP_{I'}$  do  
15   |   |   |  $N_{I''} = N_{I''} - 1$ ;  
16   |   |   | if  $N_{I''} = 0$  then  
17   |   |   |   |  $Front_{z+1} = Front_{z+1} \cup I''$ ;  
18   |   |   | end  
19   |   | end  
20   | end  
21   |  $z = z + 1$ ;  
22 until All individual is labeled with a rank;
```

---

---

**Algorithm 12: Crowding Distance**

---

```
1 foreach  $Front_q$  do  
2   | foreach  $OB^f$  do  
3   |   | Sort  $Front_q$  by ascending or descending order of the objective;  
4   |   |  $Distance_{Front_q^1} = \infty$ ;  
5   |   |  $Distance_{Front_q^{|Front_q|}} = \infty$ ;  
6   |   | foreach  $p = 2$  to  $(|Front_q| - 1)$  do  
7   |   |   |  $Distance_{Front_q^p} = Distance_{Front_q^{p-1}} + \frac{OB_{p+1}^f - OB_{p-1}^f}{OB_{|Front_q|}^f - OB_1^f}$ ;  
8   |   | end  
9   | end  
10 end
```

---

### 3.8 Basic scheme

Since the algorithm is able to solve both single objective and multi-objective case of RCPSP, the basic scheme of both are needed to be explained.

For the single objective case, preprocessing of the problem instance is performed before GA execution. The initial population is constructed by using feasible initial population function because each individual has to be feasible with respect to non-renewable resources. Precedence feasible activity list of each individual is created by either randomly or regret based biased random sampling procedure. After various mode assignment procedure proposed in this thesis, a feasible individual can be found and the algorithm makes use of tree structure to generate the remaining number of individuals. If not found, then initial population is created by random initial population function. The individuals as much as  $POP$ , which has to be even number, are created and their fitness value is assigned as described in Section 3.4. After this phase, one of the parent selection procedures are applied and each newly generated individuals are subject to one of the crossover techniques and mutation. The number of individuals existing in the population turns out to be  $2 \cdot POP$ . At this stage, the size of the population is declined to  $POP$  by using one of the population reduction techniques, which evaluates fitness values of each individual. This procedure is repeated until the given number of generations are reached.

While solving multi-objective RCPSP, some instances are not taken into account since no feasible individual can be provided with any of the mode assignment procedures while building the initial population. On the other hand, the feasible initial population can be constructed with some other instances, but it is observed that if random initial population is built with these instances, the population becomes feasible as GA proceeds. Thus, if we eliminate the instances that is not appropriate for the feasibility and whose number is very few, then feasible initial population creation procedure becomes unnecessary. In that case, random initial population can be performed. In this thesis, this becomes our case.

Therefore, it tries to create initial population by random initial population. After getting the initial population, non-dominated sorting and crowding distance procedure are applied to the population. After applying parent selection, crossover and mutation operators, the size of the population increases to  $2 \cdot POP$ . Non-dominated sorting and crowding distance are applied to this extended population. The size of the population decreases again to  $POP$  with the help of population reduction mechanism. This procedure is repeated until the prescribed number of generations is reached. It should be noted that after each generation, the solutions belonging to the non-dominated front are copied to archive.



However, it is taken into consideration that a first-rank individual is not copied, if there is already the same individual in the archive. At the end of each generation, archive is sorted with regard to the considered objectives and dominated individuals are removed from the population. At the end of GA, decision maker is presented the solutions existing in the archive.

Above mentioned scheme for multi-objective RCPSP should be accepted as basic scheme because some applications to maintain the diversity and improve the solution quality are included into the scheme. These applications will be explained in the next chapters.

## Chapter 4

### Fine-Tuning of the Parameters

#### 4.1 Commonly used performance measures in the literature

In the literature, several performance measures are developed. It should be noted that some of them require to have reference set, which is defined as the true / optimal Pareto front. On the other hand, some of them can be beneficial without having reference set.

Ballestín and Blanco [8] propose some performance measures to evaluate which algorithms can solve the multi-objective RCPSP better. Ballestín and Blanco [8] denote the set of non-dominated solutions provided by an algorithm by  $M$ , which is called the approximation set. To assess the quality of the solution, the distance between Pareto optimal front and the approximation set  $M$  needs to be measured. In case the Pareto optimal front is not known, the union of all solution sets  $M$  obtained by different algorithms is regarded as the reference set  $R$ .

##### 4.1.1 The size of the approximation set $M$

Obviously, the higher the cardinality of the approximation set  $M$  is, the more preferred it is. That is, if an algorithm can find more non-dominated solutions than another algorithm, it is said that the first one outperforms the second one. However, that is not always the case since there may be only one non-dominated solution, which makes this performance measure ineffective.

### 4.1.2 Distance from the reference set $R$

Multi-objective solution algorithms aim at generating the Pareto optimal front. Thus, it is necessary to measure the distance between the approximation set  $M$  and the Pareto optimal front. A way of performing it is to take the average of minimum distance between the approximation set and the Pareto optimal front. Czyzak and Jaszekiewicz [33] define the distance between  $M$  and the reference set  $R$  as follows:

$$D = \frac{1}{|R|} \sum_{y \in R} \min_{x \in M} \{c(x, y)\}, \quad (4.1)$$

$$\text{where } c(x, y) = \max_{k=1, \dots, p} |w_k \cdot (f_k(x) - f_k(y))|, \quad (4.2)$$

$$\text{with } x \in M, y \in R, w_k = 1/\Delta_k. \quad (4.3)$$

where  $\Delta_k$  represents the range of the  $k^{\text{th}}$  objective in the reference set.

### 4.1.3 Coverage error

As Sayin [122] proposes, the coverage error  $C$  is computed as follows:

$$C = \max_{y \in R} \{\min_{x \in M} \{c(x, y)\}\} \quad (4.4)$$

in which  $c(x, y)$  has been defined as in Equation (4.2). It is reported by Ballestín and Blanco [8] that this metric is too sensitive to the size of the sets.

### 4.1.4 Error ratio

In Van [141], a metric called error ratio is proposed and is defined as follows:

$$E = \sum_{t=1}^{|M|} \frac{e_t}{|M|} \quad (4.5)$$

where  $e_t = 0$ , if the  $t^{\text{th}}$  solution in  $M$  exists in the Pareto optimal set and it is equal to 1, otherwise. However, many algorithms can not find a large set of  $M$  or  $R$ . Thus, this metric turns out to be an inappropriate performance measure.

### 4.1.5 Hypervolume

Hypervolume has been proposed by Zitzler and Thiele [154]. This indicator measures the area between the reference point and the approximation set. In other words, for the maximization of two objectives, a non-dominated solution  $x$  forms a rectangle defined by the points  $(f_1(x), f_2(x))$  and  $(0, 0)$ . The union of the all rectangles formed by all non-dominated solutions is set as hypervolume. A larger value of hypervolume indicates better performance.

It is mentioned that this quality metric has a couple of important advantages over other metrics. One of the advantages is that each algorithm can be applied independent of other algorithms. In other words, each algorithm can be evaluated by its own without constructing the reference set. For example, let  $A$  and  $B$  become the approximation sets obtained by different algorithms. If the approximation set  $A$  dominates the approximation set  $B$  with respect to hypervolume, then  $A$  has a better performance measure than that of  $B$ .

### 4.1.6 Binary $\varepsilon$ indicator (Zitzler et al. [155])

Assuming that  $p$  objectives are tried to be minimized and all of them are positive, an objective vector  $x = (f_1^x, f_2^x, \dots, f_p^x) \in A$  is said to be  $\varepsilon$ -dominant over the objective vector  $y = (f_1^y, f_2^y, \dots, f_p^y) \in B$  written as  $x \succeq_\varepsilon y$ , if and only if

$$f_p^x \leq \varepsilon \cdot f_p^y, \quad \forall p, \quad (4.6)$$

where  $A$  and  $B$  has been defined as in Section 4.1.5. Thus, binary  $\varepsilon$  indicator (this name is written as binary additive  $\varepsilon$  indicator in Ballestín and Blanco [8]) is calculated as

$$I_\varepsilon(A, B) = \inf_{\varepsilon \in R} (\forall y \in B, \exists x \in A : x \succeq_\varepsilon y) \quad (4.7)$$

### 4.1.7 Maximum spread (Zitzler [152])

This performance indicator measures the ability of an approximation set  $M$  to diverge. Thus, it shows the size of the space covered by the approximation set. The maximum spread is defined as follows:

$$MS = \sqrt{\sum_{k=1}^p \max_{(x,y) \in M \times M} (f_k(x) - f_k(y))^2} \quad (4.8)$$

where  $p$  represents the number of objectives,  $x$  and  $y$  denote the individuals in the approximation set  $M$  and  $f_k(x)$  denotes the objective value of  $x$  in  $k^{th}$  objective.

Ballestín and Blanco [8] report that this measure should not be used alone because it does not calculate how an approximation set is close to the Pareto optimal set.

## 4.2 Other studies in the literature about performance measurement

Cheng et al. [26] evaluate the solutions with two different performance measures. Binary hypervolume indicator proposed by Zitzler and Thiele [154] and unary multiplicative  $\varepsilon$  indicator are preferred. Considering the definition of binary  $\varepsilon$  indicator explained in Section 4.1.6, unary multiplicative  $\varepsilon$  indicator is defined as  $I_\varepsilon^1(A) = I_\varepsilon(A, R)$ , where  $R$  is the union set of all non-dominated fronts obtained by different algorithms.

Fowler et al. [55] use four performance measures for evaluating the solution quality. They are visual comparison, the number of Pareto optimal solutions, the number of combined Pareto optimal solutions and Integrated Convex Preference (ICP) measures. For finding the number of combined Pareto optimal solutions, the approximate Pareto fronts found by each algorithm to be tested are unified and dominated solutions are removed from the set. In this point, each algorithm is evaluated with respect to the number of solutions of it existing in the final set. As for ICP measure, four different types of it are suggested. In order to evaluate the solution quality with ICP measure, three methods are utilized. First, ICP measure can be used directly. Thus, for comparing two algorithms, the algorithm having the least ICP measure is said to be the best. Additionally, this method can be used for fine-tuning the algorithm since it can be considered as response value during response surface optimization. Secondly, the difference between ICP measure can be used when two algorithms are compared. A third method is to use the ratio of ICP measure.  $\frac{ICP(A)}{ICP(R)}$  can be evaluated, where  $R$  is the reference set and  $A$  is one of the algorithms. In the study of Fowler et al. [55], since the reference set or true Pareto front is not known, the difference of ICP measures is preferred. At the end of the study, for a randomly selected instance, all comparison methods defined above are used and evaluated.

Cochran et al. [29] compare the algorithms by employing the following method defined by Hyun et al. [71]: Let the first algorithm be  $A$  and the second algorithm be  $B$ . Let us consider that  $A$  could find 5 non-dominated front solutions and  $B$  could find 8. Let us assume that if these two non-dominated solution sets are combined, the number of Pareto solutions becomes 6; 2 from algorithm  $A$  and 4 from algorithm  $B$ . In this case, qualitative measure of  $B$  indicates the proportion of the number of solutions obtained by  $A$  in the combined set of non-dominated solution sets. Thus, qualitative measure of the algorithm  $A$  becomes  $2/6$  and that of the algorithm  $B$   $4/6$ . Quantitative measure, on the other hand, indicates the size of the non-dominated front obtained by the algorithm. Thus, quantitative measure of the algorithm  $A$  is 5 and that of the algorithm  $B$  8.

Kılıç et al. [79] compare the performance of different algorithms with extreme hyperarea ratio (EHR) developed based on the idea of hyperarea ratio defined by Knowles and Corne [82]. In the case of the hyperarea ratio, the hyperarea bounded with non-dominated solutions obtained by an algorithm is divided by the hyperarea formed by the true Pareto solutions. However, in Kılıç et al. [79], since the true Pareto front is not known, another metric is developed called EHR. The hyperarea bounded with the solutions obtained by an algorithm is divided by the area of the rectangle having the points  $(0, 0)$  and  $(max_1, max_2)$ , where  $max_k$  represents the maximum value of the objective  $k$ . This performance measure is used for both comparing different algorithms and fine-tuning the parameters of GA.

Rojas et al. [120] adjust the GA parameters by using a well-defined experiment. They state that the parameters of GA should be determined carefully, rather than manually. Thus, they consider two different behaviors of the algorithm. Firstly, they think that the exploration ability of the algorithm is significant. That is, the ability of the algorithm to find the best solutions is taken into account. Secondly, it is claimed that the exploitation ability of the algorithm, which defines the diversity of it, is another significant aspect of the algorithm. The relevancy and the relative importance of the parameters are investigated by using a powerful statistical tool, which is analysis of variance.

### 4.3 GA parameter design in the literature

Ghoddousi et al. [56] design the parameters of GA with trial and error for a case study, which is defined and obtained for testing the algorithm. For evaluating the performance, convergence to the Pareto optimal set and maintenance of diversity in solutions are taken into account. These aspects are evaluated with graphical representation and any of the per-

formance measures is not used. For another case study, sensitivity analysis is performed for fine-tuning the parameters. Ghoddousi et al. [56] claim that after 150<sup>th</sup> generation, Pareto front solutions do not change and convergence criteria are reached.

Cheng et al. [26] make an experiment for obtaining the best parameters of the hybrid algorithm. For each parameter combination, a linear weighted sum of objectives is obtained. To generate the set of non-dominated solutions, the algorithm performs multiple runs with different weight settings.

Hanne and Nickel [60] design the parameters of multi-objective evolutionary algorithm (MOEA) with experiments. For evaluating the combination of parameters, the average relative improvement (compared with respect to the FCFS solution) for the maximum improvement (over the population) in three objective values and for three stochastic repetitions of the MOEA run are evaluated. Another approach for determining parameters is based on the idea of applying another evolutionary algorithm (EA) and this procedure is explained in Hanne [59].

Kılıç et al. [79] design the parameters of GA in the following way: Generation number and population size are related so that their product results in 50,000. In the experiment, the population size and generation number values to be tested are (100,500), (200,250), (250,200) and (500,100). The crossover and mutation rate selected from set {0.15, 0.30, 0.45, 0.60, 0.75} are determined so that their summation does not exceed 1. This pattern leads to 15 possible situations for these rates. By considering the population size and generation number values, the number of situations to be tested becomes 60. As for test data, 15 different instances are determined. Therefore, the number of experiments turns out to be 900. EHR value is used as the response variable for each experiment and ANOVA analysis is performed to understand which combination of parameters is the best. According to this analysis, none of the parameter combinations is significant at 5% significance level. Thus, the parameter combination with the best EHR value is chosen for using in GA.

#### **4.4 Getting the best parameter combination in our study**

The parameters of GA should be selected in order to improve its performance. The parameters of GA, which are crossover rate, mutation rate, population size and generation number are determined as a result of detailed fine-tuning process. This process is performed firstly for single project case of RCPSP. The similar fine-tuning process is repeated for multi-project case of RCPSP and it will be explained in Chapter ??.

Crossover Type	Roulette Wheel Selection	Binary Tournament Selection
One-Point Crossover	1	2
Two-Point Crossover	3	4
MCUOBC	5	6

Table 4.1: Design of Experiment

As for fine-tuning for single project case, five instances from 10-activity, 20-activity and 30-activity instances are randomly selected. For each instance, six different experiments varying with the type of crossover and parent selection operators are implemented with the objective combination minCMAX/maxNPV (the minimization of Cmax and the maximization of Npv). These experiments are shown in the Table 4.1:

As can be seen from Table 4.1, each cell represents an experiment for an instance. In other words, for each of 15 instances, all of these six experiments are implemented. During each experiment, an instance is solved with varying parameter values. The range for crossover rate, mutation rate, population size and generation number are given in the following list:

**Crossover rate** [0.6 , 1.0] increasing in increments of 0.1

**Mutation rate** [0.01 , 0.25] increasing in increments of 0.04

**Population size** [20 , 100] increasing in increments of 20

**Generation number** [25 , 150] increasing in increments of 25

For each combination of these parameter values (for example, crossover rate: 0.8, mutation rate: 0.2, population size: 100, generation number: 125), each instance is solved five times to reduce the undesired effect of randomness. This number might be larger than five, but even in this case, the time spent for the experiment becomes huge. In each solution, the performance of the provided non-dominated solution set is calculated by using some performance measures.

In Section 4.1, seven performance measures in the literature are explained. Among these, four of them require to find the reference set, which may represent the true / optimal Pareto set. However, since the reference set is unknown in our case, three of the performance measures, which do not require to have the reference set are preferred for measuring the performance of the parameter combinations. In fact, the reference set could be constructed by taking the union of approximate Pareto fronts obtained by all parameter combinations. However, in this case, the obtained reference set might have been biased



because the contribution of the high population size and high generation number to the reference set would be most likely very large. Thus, the results would be in favor of high population size and generation number. In order to avoid this kind of situation, the performance measures, which do not require to have the reference set are preferred.

Hypervolume indicator defined by Zitzler and Thiele [154] (see Section 4.1.5) measures the rectangular area constructed by the approximate Pareto solution points and a reference point. The larger this area is, the better the corresponding parameter combination shows performance. The following figures represent the hypervolume indicator:

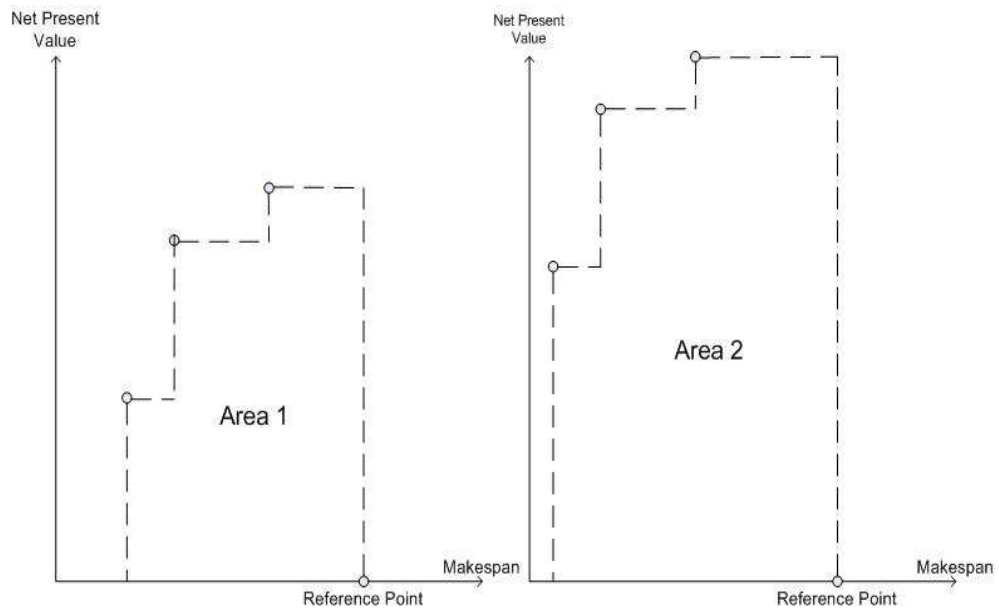


Figure 4.1: Hypervolume Indicator for Two Different Cases

The first case of Figure 4.1 represents the area, Area 1, constructed by three approximate Pareto solutions and a selected reference point. In the second figure, the same approximate Pareto solutions form larger area, Area 2. If it is desired to compare which one is better in terms of hypervolume, it should be stated that the second one is better because Pareto solutions of it seems to be closer to unknown true Pareto front. It should be stated that the objectives in y-dimension and x-dimension are maximized and minimized, respectively.

Maximum spread developed by Zitzler [152] (see Section 4.1.7) shows how well Pareto solutions spread. For measuring this value, Equation (4.8) in Section 4.1.7 is calculated. However, the pure difference is not preferred in our study. That is, differences are scalarized with a meaningful value. Thus, with this equation, the largest difference between the solutions for each objective are found.

The number of approximate Pareto solutions is another performance measure that does not require to use the reference set. It is obviously a superiority to obtain more Pareto solutions. Actually, this performance measure is defined by Hyun et al. [71] as quantitative measure.

Hypervolume, maximum spread and the number of non-dominated solutions are used as performance measures. As stated before, an instance in an experiment is solved with a parameter combination (denoted by  $i$ ) five times (each of them is denoted by  $j$ ) and hypervolume indicator (denoted by  $H_{ij}$ ), maximum spread (denoted by  $M_{ij}$ ) and the number of non-dominated solutions (denoted by  $Q_{ij}$ ) are obtained. For fine-tuning procedure, the maximum value of the obtained performance measures should be provided with the equations (4.9), (4.10) and (4.11).

$$Max_H = \max(H_{ij}), \forall i, j \quad (4.9)$$

$$Max_M = \max(M_{ij}), \forall i, j \quad (4.10)$$

$$Max_Q = \max(Q_{ij}), \forall i, j \quad (4.11)$$

For a parameter combination, the average of the performance measure values are computed with the list of equations (4.12), (4.13) and (4.14).

$$\overline{H}_i = \frac{\sum_{j=1}^5 H_{ij}}{5} \quad \forall i, \quad (4.12)$$

$$\overline{M}_i = \frac{\sum_{j=1}^5 M_{ij}}{5} \quad \forall i, \quad (4.13)$$

$$\overline{Q}_i = \frac{\sum_{j=1}^5 Q_{ij}}{5} \quad \forall i, \quad (4.14)$$

For the fine-tuning procedure, response surface optimization method (RSOM) (Myers and Montgomery [105]) is utilized. It is a well known statistical method for optimizing multiple response / output variables when there exist multiple input variables. In our case, response variables turn out to be  $\overline{H}_i$ ,  $\overline{M}_i$  and  $\overline{Q}_i$ . Input variables becomes the combinations of crossover & mutation rates, population size and generation number. RSOM

requires to set the lower and upper bounds for each of the performance measures. However, in our problem, since we do not have a determined lower and upper bounds, these are set to 0 and the maximum value of them calculated before with the equations (4.9), (4.10) and (4.11). Therefore, desirability function of the  $\overline{H}_i$ ,  $\overline{M}_i$  and  $\overline{Q}_i$  are defined as in Equation (4.4):

$$d_{\overline{H}_i} = \begin{cases} 0, & \text{if } \overline{H}_i < Lower \\ \left(\frac{\overline{H}_i - Lower}{Upper - Lower}\right)^s, & \text{if } Lower < \overline{H}_i < Upper \\ 1, & \text{if } \overline{H}_i > Upper \end{cases}$$

where  $s$  denotes how strictly this function should be. That is, if  $s = 1$ , the function increases linearly from the lower bound to the upper bound. Otherwise, it forms a convex function, if  $s < 1$  and concave function, if  $s > 1$ . It should be noted that the Equation (4.4) is showed in the case that  $\overline{H}_i$  is desired to be maximized. If it is to be minimized, then the Equation (4.4) should have the form of

$$d_{\overline{H}_i} = 0, \text{ if } \overline{H}_i > Upper \quad (4.15)$$

and the Equation (4.4) should be

$$d_{\overline{H}_i} = 1, \text{ if } \overline{H}_i < Lower \quad (4.16)$$

The desirability values of  $\overline{M}_i$  and  $\overline{Q}_i$  must be computed with the same Equation (4.4). At the end, RSOM proposes that the overall desirability of the parameter combination  $i$  is calculated with

$$D_i = (d_{\overline{H}_i} \cdot d_{\overline{M}_i} \cdot d_{\overline{Q}_i})^{\frac{1}{3}} \quad (4.17)$$

The parameter combination, which has  $\max(D_i)$  is selected for the corresponding instance and experiment. Interesting readers are referred to Table A.1 in Appendix A.

At this point, each combination of crossover and parent selection types (experiment) has 15 parameter combinations which are determined as the best. Five of these parameter combinations are for 10-activity, five of them are for 20-activity and the remaining ones are for 30-activity instances. For each instance set, which means the instances having the same number of activities, the best parameter combination should be selected among five combinations. In order to select, the closest parameter combination to another combina-

Instance	Population Size	Generation Number	Mutation Rate	Crossover Rate	Desirability	
10 Activity	1	60	25	0.25	1.00	0.44
	2	60	25	0.25	0.80	0.83
	3	80	25	0.25	0.60	0.56
	4	80	25	0.21	0.60	0.59
	5	100	25	0.25	0.80	0.68
20 Activity	1	80	125	0.17	1.00	0.40
	2	100	50	0.17	0.80	0.39
	3	100	125	0.21	0.90	0.45
	4	100	150	0.25	0.90	0.41
	5	100	75	0.17	0.80	0.40
30 Activity	1	100	150	0.05	0.80	0.34
	2	100	150	0.17	1.00	0.31
	3	80	50	0.21	0.70	0.26
	4	100	125	0.09	0.70	0.32
	5	100	150	0.05	0.60	0.22

Table 4.2: The Result of the Experiment 1

tions is selected. The closeness is defined as having the minimum deviation from other parameter combination for population size, generation number, crossover and mutation rates.

The mathematical calculation of the notion closeness is given in Equation (4.18):

$$\begin{aligned}
Closeness_k = & \frac{\sum_{m=1}^5 |Pop_k - Pop_m|}{100} + \frac{\sum_{m=1}^5 |Gen_k - Gen_m|}{150} \\
& + \frac{\sum_{m=1}^5 |Mut_k - Mut_m|}{0,25} + \frac{\sum_{m=1}^5 |Cross_k - Cross_m|}{1}
\end{aligned} \tag{4.18}$$

where  $k$  denotes a parameter combination,  $Pop_k$ ,  $Gen_k$ ,  $Mut_k$  and  $Cross_k$  denotes the population size, generation number, mutation and crossover rates belonging to parameter combination  $k$ . The denominators of the Equation (4.18) represents the maximum value the range experienced in this study.

For instance, Table 4.2 shows the result of the Experiment 1. To be clearer, the absolute deviation of the first parameter combination from the other 10-activity parameter combinations is calculated as 2.16.

This process is repeated for all the parameter combinations within the same set of

instances. Thus, for each experiment, three different parameter combinations belonging 10-activity, 20-activity and 30-activity instances are obtained. During determining which parameter combinations are the closest to the other ones, if there is an equality for minimum closeness values, the parameter combination with the highest desirability value is selected. One can observe the results of this operation in Table A.2 of Appendix A.

## **4.5 Selecting the best operant combination**

As can be seen easily in Table 4.1, this study has three different crossover and two different parent selection techniques. Thus, the best combination among them needs to be determined.

For doing this, GA is performed for all 10-activity, 20-activity and 30-activity instances varying with operant combinations. Thus, six different approximate Pareto fronts are provided for an instance. The most important issue is to determine which operant combination (approximate Pareto front) is the best among others. In order to answer this question, performance measures, which are hypervolume, maximum spread and the number of non-dominated solutions on the approximate Pareto front are used.

For an instance, hypervolume, maximum spread and the number of approximate Pareto solutions are calculated for each operant combination. The desirability and the degree, which represents the rank of each operant combination in terms of desirability are recorded. This operation is repeated for all instances.

Following this point, the average of desirability values is calculated for each operant combination. In addition, the average of degree values and the frequency, which shows how many times an operant combination is the first are also calculated. Observing all of those statistical results, it is concluded that one-point crossover and roulette wheel selection mechanism outperforms other combinations. (see Table A.3 in Appendix A)

## **4.6 Fine-tuning process with multi-project instances**

The experiment explained until this point is conducted by using single projects whose maximum number of activities is 30. Therefore, while using multi-project, the population size and generation number value determined as the best might not be efficient since the number of activities in multi-project instances can be very large. Thus, a similar and small fine-tuning experiment may be appropriate for multi-project case.

The experiment procedure becomes nearly the same with the previous one, but some differences are observed in the current experiment. Firstly, population size and generation number values to be experimented are not steady and these values are functions of the number of activities,  $N$ . That is, if the number of activities  $N$ , then the population size values to be experimented are  $[0.75N, N, 1.25N, 1.5N]$  and the generation number values are  $[N, 1.5N, 2N, 2.5N, 3N]$ . Secondly, crossover and mutation rate values are taken as constant because the time spent for the experiments would become tremendous, if those were included in the experiment. Thus, mutation and crossover rate determined as the best for 30-activity instances while applying one-point crossover and roulette wheel selection are selected as the constant values. Those values are 0.13 and 1 for mutation and crossover rate, respectively. (see Table A.2 in Appendix A).

The other difference regarding the current fine-tuning procedure is that crossover and parent selection mechanisms are not tested. Considering the results of the previous experiment, it can be concluded that one-point crossover and roulette wheel selection turn out to be the best while being applied together. Thus, it is thought to be enough to consider only these two mechanisms.

Finally, the previous fine-tuning experiment is conducted by considering only the minimization of  $C_{max}$  and the maximization of  $N_{pv}$ . However, in the current testing, all of the objective combinations implemented in GA are included in the experiment procedure. Thus, at the end of the process, the best population size and generation number values are obtained for each objective combination. At this point, it should be noted that hyper-volume indicator can easily be calculated when GA works with bi-objective. However, if triple-objective is the case, the calculation of this performance measure is challenging. In the literature, several methods are proposed for computing it efficiently and correctly. In this thesis, the computation proposed by Fleischer [53] is applied.

As for the instance files used during the experiments, the multi-project instances having 10 projects are preferred. Each project can have 10, 12 or 14 activities. Thus, total number of activities can take the value 100, 120 or 140. For 10 projects with 10 activities, three different instances varying depending on due date, resource capacity and lump sum payment difficulties are selected. In other words, 10 projects having 10 activities whose due date difficulty is in medium level, 10 projects having 10 activities whose resource capacity difficulty is in medium level and 10 projects having 10 activities whose lump sum payment difficulty is in medium level are selected. With the same consideration for 120 and 140 activities, total number of instances used in the experiments turns out to be nine.

The results of the fine-tuning procedure can be seen in Table B.1 in Appendix B. In

Objective Combinations	Coefficients for	
	Population Size	Generation Number
minCMAX/maxNPV	1.25	2.50
maxNPV/minMFT	1.25	3.00
maxNPV/minMWT	1.00	2.50
maxNPV/minMCT	1.25	3.00
minCMAX/maxMNPV/minRUD	1.00	2.50
minCMAX/maxMNPV/maxMO	1.00	2.50

Table 4.3: The Best Parameter Combinations

that table, the best population size and generation number coefficients for each instance and for each objective can be observed. For a general understanding, the detailed results of the experiment can be seen in Table B.2 of Appendix B. In that table, the best five population size and generation number coefficients exist with their desirability values for each instance and for each objective.

At the end of the experiment, the best parameter combination should be selected for each objective. The selection is achieved by computing the distance between parameter combinations and choosing the closest combination to others. In Table B.3 in Appendix B, the closeness and desirability values for each instance and for each objective can be seen. By selecting the closest parameter combination, the following population size and generation number coefficients in Table 4.3 are determined for the objectives:

In this table, the objectives represented in the first column are explained in Chapter 3.

# Chapter 5

## Divergence Applications and Local Searches

### 5.1 Divergence applications

Incorporating divergence capability into GA is one of the more important issues to be considered when implementing the algorithm. When divergence is not implemented, the algorithm can converge prematurely and it might lead to low quality solutions. In this thesis, four different divergence applications are implemented. These can be performed either independently or in a hybrid way.

#### 5.1.1 Entropy-based divergence application

As stated before, the individual is represented with two lists, which are the precedence feasible activity list and mode assignment list. Thus, in any generation, some individuals can resemble to each other in terms of activity positions and the kind of modes assigned to the activities. Entropy-based divergence application depends on this consideration.

Nsakanda et al. [112] deal with Cell Formation Problem (CFP) and GA is preferred as solution methodology. The genes in an individual represent which machine is assigned to which cell. For avoiding the premature convergence, the authors propose an equation for measuring the structural similarities of the individual existing in a population.

1	1	3	3	2	2	3
---	---	---	---	---	---	---

Figure 5.1: An Example of an Individual for CFP

For instance, in Figure 5.1, machine 1 and machine 2 are assigned to the first cell.



Machine 5 and machine 6 are assigned to cell 2 and the cell 3 has the machine 3, 4 and 7. In fact, the major purpose of the study by Nsakanda et al. [112] is to show that entropy-based measure is not suitable for measuring the population diversity for CFP. Thus, a new measure called distance-based measure is proposed by the authors. However, in this thesis, entropy-based measure is focused because it is considered to be appropriate for the individual representation in RCPSP.

Entropy-based measure is implemented for CFP in the following way:

- $m$ : The number of machines to be assigned
- $U_i$ : The contribution of the machine  $i$  to the population diversity
- $v_{ij}$ : The number of chromosomes in which machine  $i$  is assigned to cell  $j$  in the current population (generation)
- $POP$ : Population size
- $c$ : The number of cells
- $DG$ : Convergence measure

Then, the Equation (5.1) gives the entropy-based measure. According to this equation,  $DG$  converges to zero as the structural similarities of the individuals in a population increase.

$$DG = \frac{\sum_{i=1}^m U_i}{m} \quad (5.1)$$

$$U_i = \frac{-\sum_{j=1}^c \frac{v_{ij}}{POP} \cdot \log \frac{v_{ij}}{POP}}{\log c} \quad (5.2)$$

An entropy-based measure for individual representation in our case can be developed as follows: Let the activities in our problem replace the machines to be assigned in CFP and let the positions in an individual replace cells in CFP. Thus, the same equations (5.1) and (5.2) can be used for RCPSP as well.

Furthermore, another entropy-based measure can also be developed for the mode assignment list. If the activities replace the machines and the modes replace the cells, the

same equation shows how well the population has diversity in terms of modes. Therefore, by focusing on both the activities and the modes, two distinct entropy-based measures, which are called activity entropy and mode entropy can be proposed.

For simplicity, an example about how these measures can be calculated is given in Figure 5.2.

2	3	4	1	5	6	Individual 1
1	1	2	2	3	3	
3	4	2	1	6	5	Individual 2
2	1	3	2	1	1	
3	2	6	4	5	1	Individual 3
2	3	2	1	2	2	
5	1	6	4	3	2	Individual 4
2	2	3	3	3	1	
5	3	4	1	6	2	Individual 5
3	2	2	1	1	1	

Figure 5.2: An Example Calculation of Divergence Measure for RCPSP

In this example, the population has five individuals. In each individual, the first and the second lists show the precedence feasible activity list and mode assignment list, respectively. According to this figure, the contribution of the activity 2 to the population diversity is calculated in Equation (5.3).

$$U_2 = \frac{3 \cdot \left(\frac{1}{5} \cdot \log \frac{1}{5}\right) + \left(\frac{2}{5} \cdot \log \frac{2}{5}\right)}{\log 6} \quad (5.3)$$

The activity 2 is assigned to the positions 1, 2 and 3 once. For the position 6, it is assigned to this position twice. With the same consideration, the contribution of the other activities to the population diversity can be calculated and the averages of  $U_i$  values shows the overall diversity for activity entropy.

As for modes, the mode entropy measure can be calculated for activity 2 as in Equation (5.4).

$$U_2 = \frac{\frac{3}{5} \cdot \log \frac{3}{5} + \frac{2}{5} \cdot \log \frac{2}{5}}{\log 3} \quad (5.4)$$

The activity 2 is assigned to mode 3 twice and to mode 1 three times. By repeating this procedure for all activities, the overall mode diversity for mode entropy can be provided.

While controlling the diversity of the population, both measures can be checked either independently or by converting them into one measure. In this thesis, those are checked independently because the precautions taken are different.

The threshold values for both entropy measures are determined and if one of them is less than this threshold value, then its own precaution is applied. For activity entropy measure, if the divergence value is less than the determined threshold value, then the individuals are sorted in the order of activity position similarity. That is, the individual that is the most similar to each other becomes the first. At this point, predetermined number of individuals from the beginning of this list are removed from the population. Instead of them, randomly generated individuals are replaced.

For mode entropy measure, if the divergence value is less than the threshold value, then a list that sorts the individuals in terms of mode assignment similarity is created. The predetermined number of individuals starting from the beginning of this list are modified by changing the modes of the activities.

In this kind of divergence application, the way that the threshold values for activity and mode entropy are determined is obviously critical to the quality of GA solutions. Moreover, it should also be considered as important to determine how many individuals are removed from the population. Our purpose is to determine these values by some experiments.

### **5.1.2 Objective value-based divergence application**

In addition to the structural similarities of the individuals, similarities in objective value should also be checked because both similarities may behave differently. In other words, while the individuals have not similarity in terms of entropy-based measure, they might be very similar to each other in terms of objectives, or vice versa. Thus, objective value-based divergence application should also be developed.

While determining whether an individual is similar to another, the Euclidean distance is used. With the help of its definition, the distance between two objective vectors can be computed easily. The procedure of controlling the objective value-based diversity is explained below:

- $p$ : The number of objectives
- $x, y$ : Objective vectors (individuals)
- $f_k(x)$ : The  $k^{th}$  objective value of  $x$

- $\max_k$ : The maximum  $k^{th}$  objective value in current population

The Euclidean distance between two individuals is computed using Equation (5.5).

$$ED_{x,y} = \sqrt{\left(\frac{f_1(x) - f_1(y)}{\max_1}\right)^2 + \left(\frac{f_2(x) - f_2(y)}{\max_2}\right)^2 + \dots + \left(\frac{f_p(x) - f_p(y)}{\max_p}\right)^2} \quad (5.5)$$

For all pairs of objective vectors existing in the current population, the Euclidean distances between them are computed and the  $\Gamma$  value is provided by the following Equation (5.6).

$$\Gamma = \frac{\sum_{(x,y) \in Population} ED_{x,y}}{POP} \quad (5.6)$$

As in entropy-based divergence application,  $\Gamma$  value must be compared to a threshold value and if  $\Gamma$  is less than it, then it should be accepted as a sign of convergence in terms of objective similarity. In this case, each individual is compared to other individuals by observing the distance between them. If the distance is less than  $\Gamma$ , then the Euclidean distance frequency (EDF) of the corresponding individual is incremented by one. The Algorithm 13 summarizes this process:

---

**Algorithm 13:** Determining The Closest Individuals

---

```

1 foreach Individual  $x \in Population$  do
2   foreach Individual  $y \in Population$  do
3     if  $ED_{x,y} < \Gamma$  then
4        $EDF_x = EDF_x + 1;$ 
5     end
6   end
7 end

```

---

After completing this procedure, the predetermined number of individuals are removed from the population starting with the individual whose  $EDF$  value is the highest. In case of equality, the individual whose rank value is the worst is removed from the population. In place of removed individuals, randomly generated individuals are inserted.

Obviously, the threshold value and the number of individuals to be removed should be selected carefully because these make an impact on the process of GA. Our purpose is to determine these values by some experiments.

### 5.1.3 Grid-based divergence application

The other divergence application, which is based on objective space is grid-based. In other words, objective space is divided into grids in a rational way. In that case, if a grid has two or more individuals, then it is accepted as a warning that these individuals are similar to each other. Thus, it might be beneficial to remove all of them but one.

This divergence application is developed in case objective value-based divergence may not be effective. It should be noted that the decision about this choice should be made by several experiments.

Since grid-based divergence is checked in every generation, the reference lines or grid borders should be modified as generations proceed. That is, if the reference lines are not changed through generations, the natural convergence of GA is not taken into account because the reference lines used for the initial population might become very different than those used for any following generation.

Grid-based divergence application works as in Algorithm 14:

- $RF_k(l)$ : The  $l^{th}$  reference line for  $k^{th}$  objective
- $f_k(t)$ : The  $k^{th}$  objective value of  $t^{th}$  individual in the current population

---

#### Algorithm 14: Grid-Based Divergence Procedure

---

```

1 foreach Objective  $k$  do
2    $Sum = 0$ ;
3    $Average = 0$ ;
4    $l = 1$ ;
5   Sort the population in increasing order of  $k$ ;
6   for  $t = 2$  to  $POP$  do
7      $Sum = Sum + (f_k(t) - f_k(t - 1))$ ;
8   end
9    $Average = \frac{Sum}{POP-1}$ ;
10   $RF_k(l) = f_k(l)$ ;
11  repeat
12     $RF_k(l) = RF_k(l - 1) + Average$ ;
13     $l = l + 1$ ;
14  until  $RF_k l = f_k(| POP |)$ ;
15 end

```

---

Thus, for each objective, reference lines are determined by this procedure. In other words, for the same  $l$  value, the point  $(RF_1(l), RF_2(l), \dots, RF_p(l))$  represents a corner

point for grid in the  $p$ -objective space. Afterwards, grid-based divergence application checks all of the individuals and removes all individuals, which exist in the same grid except for one. Removing process starts with the worst individuals in terms of rank values. Thus, single remaining individual becomes the best individual among those existing in the same grid. If there is an equality in rank values, then randomly selected individual continues occupying grid. Instead of removed individuals, randomly generated individuals are inserted to the population.

It is clear that the way of deciding the width of the reference lines is a strategy, which should be adjusted carefully. That is, the average value of the summations of differences may not be preferred. Instead, if the first quartile of the summation is applied, then the width of the reference lines become more narrow or if third quartile of the summation is implemented, then they become wider.

#### **5.1.4 Archive-based convergence check**

Beside all the divergence applications mentioned above, a simple and not time-consuming convergence check is also applied in our study. According to this kind of application, the archive is checked in every generation. If no individual is added to the archive through predetermined number of generations, then this is accepted as a sign of convergence because GA can not find any better solution than the existing ones in the archive.

## **5.2 Local searches**

Local searches can be regarded as a divergence application because the algorithm that prematurely converges can be diverged with the help of them. In addition to divergence, improvement is another important aspect of local search procedures. In this thesis, two different local search techniques are applied. Both can be applied while GA continues proceeding or when GA stops. In other words, any population or the last non-dominated solutions, which will be presented to the decision maker can be subjected to local search.

### **5.2.1 Backward and forward pass procedure**

Backward and forward pass (BFP) procedure aims at assigning new starting and finishing times for the activities without changing the modes so as to improve the solution quality.

A pass in this procedure can be defined as shifting all of the activities to right or to left by using the slacks. The first pass to be applied is backward pass, which means

shifting all of the activities to right by considering some constraints. Before starting the shifting process, the activities are sorted in decreasing order of their finishing times. Afterwards, the activities are shifted as much as possible to right starting the first activity in this order. The constraints taken into account are precedence relations and renewable resource capacities. The non-renewable resource capacities are not checked because the modes are not changed.

The second iteration is forward pass, which operates by shifting all activities to left by taking into account the same constraints. First, the activities are sorted in increasing order of their starting times. By following the same procedure with backward pass, the activities in an individual are assigned new starting times and finishing times (see Figure 5.3).

However, the procedure does not only implement the BFP, but also some additional operations. Firstly, as stated before, it applies backward pass on the individual. At this point, it is possible that the starting time of the dummy source activity is larger than zero. In that case, the starting and finishing times of all activities are decreased by the starting time of the dummy source activity. Thus, a new individual is provided ( $I_1$ ), but original individual ( $I_0$ ), which is the output of the backward pass is kept distinctly.

By applying forward pass on  $I_0$ , a new individual is created ( $I_2$ ). After implementing backward pass on  $I_0$  again, the starting time of the dummy source activity is checked and if it is larger than zero, then the starting and finishing times of all activities are decreased by that value ( $I_3$ ). Afterwards,  $I_0$  is subjected to forward pass again ( $I_4$ ).

At each pass, the objective values and renewable resource profiles are recalculated. Starting from the first individual  $I_1$ , an individual list is formed that includes all non-dominated solutions. BFP procedure is terminated until this list can not be renewed through predetermined number of passes. At the end, all of the non-dominated solutions are accepted as the improved form of the initial individual. The flow chart of the general scheme can be seen in Figure 5.4.

## 5.2.2 Simulated annealing

Another local search application can be implemented by simulated annealing (SA). SA is observed to be an effective heuristic for this kind of problems such as RCPSP (Hartmann and Kolisch [67]). In our procedure, the neighborhood search is performed through changing the position of an activity and changing the mode of another activity.

While changing the position, after randomly selecting an activity, the positions of the

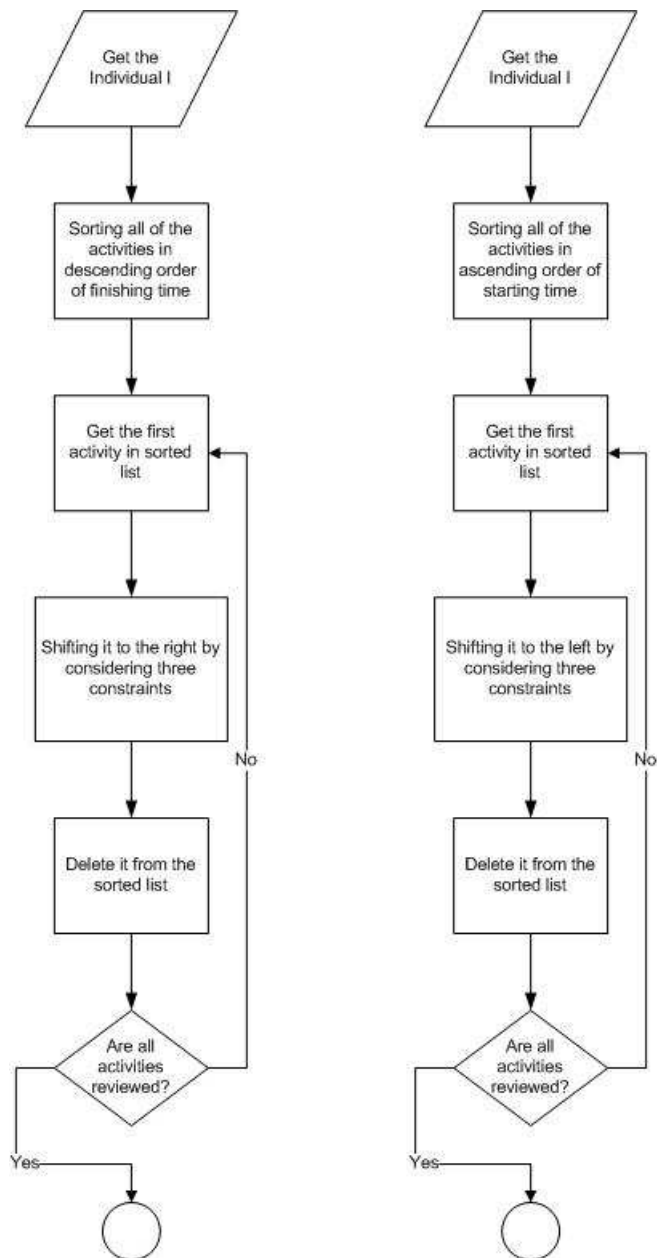


Figure 5.3: BFP Procedure



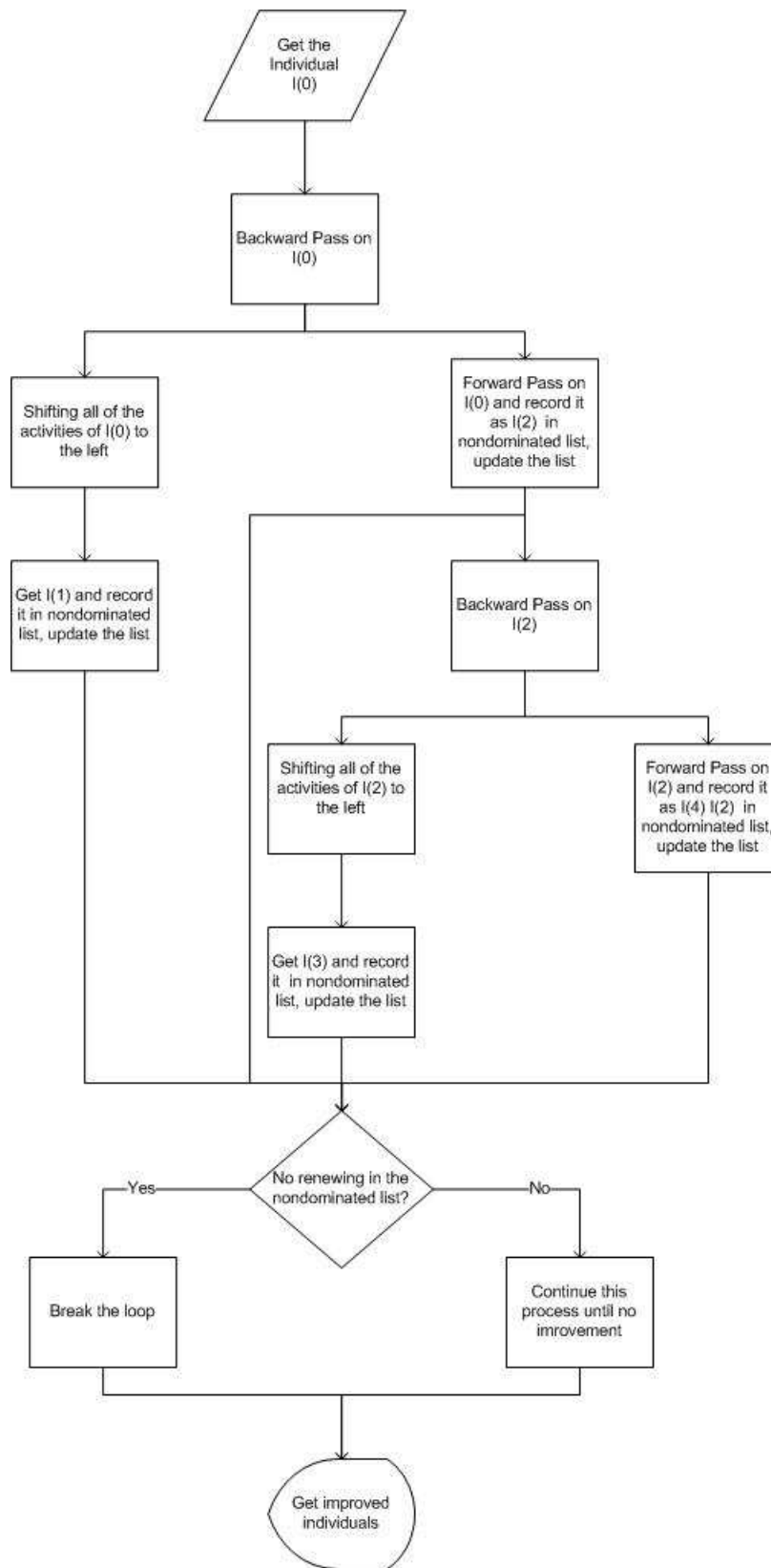


Figure 5.4: General Framework of BFP

last predecessor  $lp$  and the first successor  $fs$  of it are determined. Then, this activity is inserted into a position between  $lp$  and  $fs$ . Remaining activities existing between  $lp$  and  $fs$  are shifted as a block to prevent the precedence feasibility.

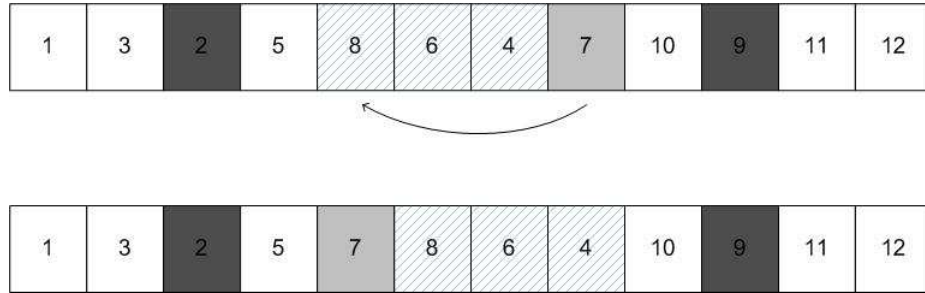


Figure 5.5: An Example of Activity Insertion

For instance, in Figure 5.5, activity 7 is selected randomly. The  $lp$  and  $fs$  of it are the activities 2 and 9. Between the activities 2 and 9 (i.e.,  $lp$  and  $fs$ , respectively), activity 8 is selected randomly in order to insert activity 7 before activity 8 in the sequence. In this case, the block of activities 8, 6 and 4 are shifted to right together to preserve the precedence feasibility.

After finding a neighbor called the candidate solution, it is compared to the current solution with respect to the objectives in order to decide whether the candidate solution must be accepted as the current solution. As SA proposes, some worse candidate solutions can be accepted as current solutions with a probability. As the iterations proceed and temperature decreases, acceptance probability of worse individuals decreases. The probability of acceptance is calculated depending on the difference between the objective values of the current and candidate solutions.

While comparing, there exist several situations, all of which must be explained in detail. Let  $x$  be the current solution and  $y$  be candidate solution. Additionally, let  $e^{-\frac{\delta}{T}}$  be the probability of accepting  $y$ , where  $\delta$  represents the difference between objective values of  $x$  and  $y$  and  $T$  denotes the temperature in the current iteration. Then, following definitions can be made:

- $p$ : Number of objectives
- $k$ :  $k^{th}$  objective
- $f_k(x)$ : The  $k^{th}$  objective value of  $x$
- $f_k(x) \triangleright f_k(y)$ : The  $k^{th}$  objective value of  $x$  is better than that of  $y$

---

**Algorithm 15:** Acceptance Procedure When  $p = 2$ 

---

```
1  $r$ : A decimal random value between  $[0,1]$ ;  
2 if  $f_1(y) \triangleright f_1(x)$  and  $f_2(y) \triangleright f_2(x)$  then  
3 |   Accept  $y$  as current solution;  
4 end  
5 if  $f_1(y) \triangleright f_1(x)$  and  $f_2(x) \triangleright f_2(y)$  then  
6 |   Generate  $r$ ;  
7 |    $\delta = |f_2(x) - f_2(y)|$ ;  
8 |   if  $e^{-\frac{\delta}{T}} > r$  then  
9 | |   Accept  $y$  as current solution;  
10 | end  
11 end  
12 if  $f_1(x) \triangleright f_1(y)$  and  $f_2(y) \triangleright f_2(x)$  then  
13 |   Generate  $r$ ;  
14 |    $\delta = |f_1(x) - f_1(y)|$ ;  
15 |   if  $e^{-\frac{\delta}{T}} > r$  then  
16 | |   Accept  $y$  as current solution;  
17 | end  
18 end  
19 if  $f_1(x) \triangleright f_1(y)$  and  $f_2(x) \triangleright f_2(y)$  then  
20 |   Generate  $r$ ;  
21 |   Generate a random integer  $r' \in [1, 2]$ ;  
22 |    $\delta = |f_{r'}(x) - f_{r'}(y)|$ ;  
23 |   if  $e^{-\frac{\delta}{T}} > r$  then  
24 | |   Accept  $y$  as current solution;  
25 | end  
26 end
```

---

As can be seen in Algorithm 15, if both objectives of  $y$  are better than those of  $x$ , then it is accepted as the current solution without calculating the acceptance probability. If one of the objectives of it is worse, then it is accepted as current solution, if acceptance probability computed depending on the difference between these objectives is bigger than a randomly generated decimal value. If both objectives are worse, then one of them is selected randomly and the same acceptance procedure is applied.

There also exist three objectives considered simultaneously in this thesis. In that case, above procedure becomes longer because the number of situations to be investigated increases. If one of the objectives or two of the objectives are worse, then the same procedure is applied during accepting the candidate solution. If all of the objectives are worse, then one of them is selected randomly and acceptance probability is provided depending on that objective.

The basic scheme of the simulated annealing procedure is presented in Algorithm 16.

---

**Algorithm 16:** Basic Scheme of Local Search Using Simulated Annealing

---

```

1  $T_0$ : Initial temperature;
2  $T = T_0$ : Temperature varying in each iteration;
3  $H$ : The number of iterations;
4  $\alpha$ : A decimal value in  $[0,1]$ 
5  $\beta$ : Termination condition
6 repeat
7    $H = \lceil \frac{1}{T} \rceil$ ;
8   while  $H > 0$  do
9     Find a neighbor ( $y$ ) of current solution;
10    Accept or reject  $y$  as current solution;
11     $H = H - 1$ ;
12  end
13   $T = T \cdot \alpha$ ;
14 until  $T \leq \beta$ ;
```

---

# Chapter 6

## Computational Study

### 6.1 Implementation with test instances

In order to evaluate the performance of GA, the algorithm is tested with different multi-project test instances. In addition to using the instances generated in the study of Can and Ulusoy [20], new multi-project test instances generated in this thesis are also utilized. The performance of GA implemented can be compared to that of Can and Ulusoy [20] because the same instances are solved.

#### 6.1.1 Implementation with the test instances generated in Can and Ulusoy [20]

Three problem sets denoted by A, B and C are created to represent a variety of different environmental factors. Problem set A is formed to analyze the effect of resource based factors while fixing other factors. Set A includes multi-project cases with the same number of projects and the same number of activities but different resource requirements and resource availability levels. Each instance includes 14 projects consisting of 10 activities. Problem set B consisting of 84 instances focuses on the effects of different number of projects and activities. In these multi-project instances, three levels are set for the number of projects and seven levels are set for the number of activities. In problem set C, a multi-project environment that is heterogeneous in terms of project sizes, is emphasized by grouping projects consisting of different number of activities resulting in 27 instances. The objective combination studied is minCMAX/maxNPV. First, NSGA-II is employed to solve these sets of test problems. BFP is applied in two different modalities. In one

modality, it is applied on the archive of non-dominated set of solutions obtained at the end of NSGA-II and is designated as BFP in the final stage. As an improvement routine, BFP is also applied in the intermediate stages, which is designated as BFP in the intermediate stages.

As for intermediate application of BFP, it should be determined in which condition BFP is applied. One possible solution is to take into account the number of generations in which no better solution is inserted into the archive. If a threshold value is determined for the discontinuation of the archive, the algorithm implements the BFP procedure on the population and convergence of the algorithm is tried to be handled by this way.

In order to assign the best threshold values for the instance sets A, B and C taken from Can and Ulusoy [20], the algorithm is performed in advance to observe the behavior of the archive since the archive behaves differently due to the fact that each instance has different difficulty. During pre-solving the instances, the generation numbers in which the archive reaches to discontinuation levels 5, 7, 9, 11, 13, 15, 17, 19, 21, 23 and 25 are recorded. For example, A11\_11 instance has reached to discontinuation level 5 at first when the generation number is 203. During solving the same instance, the algorithm has stopped before reaching to discontinuation level 15. With the same considerations, all of the multi-project instances in A, B and C sets are pre-solved and corresponding generation numbers are recorded.

As state before, there are 81 instances in set A, 84 instances in set B and 27 instances in set C. After recording the corresponding generation numbers, the instances existing in the same instance set are separated into the subgroups. Finally, there are 9 subgroups in set A, 21 subgroups in set B and 3 subgroups in set C. Actually, the instances in the same subgroups have similar difficulty. This difficulty level is arranged in the phase of data generation in Can and Ulusoy [20]. For each subgroup, the average value of the recorded generation numbers for the same discontinuation level are calculated. Discontinuation level whose average recorded generation number is larger than the half of the bound of generation number is determined as the threshold value.

It is observed that the algorithm can not improve the solution quality after applying a couple of BFP. In this case, the algorithm can be stopped and non-dominated solutions can be presented to the decision maker. It is determined in this thesis that if the algorithm applies BFP five times and can not find better solutions than current solutions during each application, then the algorithm is stopped. This value is represented as stoppage in the tables. Table 6.1 shows the determined threshold values and stoppage values for the instance subgroups.

Instance	Threshold	Stoppage	Instance	Threshold	Stoppage	Instance	Threshold	Stoppage
A11	13	5	B1014	9	5	B1530	5	5
A12	21	5	B1016	15	5	B2010	19	5
A13	21	5	B1018	19	5	B2012	23	5
A21	5	5	B1020	15	5	B2014	11	5
A22	17	5	B1030	5	5	B2016	13	5
A23	19	5	B1510	13	5	B2018	5	5
A31	5	5	B1512	23	5	B2020	7	5
A32	17	5	B1514	5	5	B2030	5	5
A33	19	5	B1516	15	5	C1	7	5
B1010	5	5	B1518	7	5	C2	15	5
B1012	5	5	B1520	7	5	C3	11	5

Table 6.1: Threshold and Stoppage Values for BFP Procedure

Table 6.2 presents the explanations of various abbreviations used in the following tables of this chapter.

The instance sets A, B and C are solved with GA for minCMAX/maxNPV by applying BFP both in the intermediate stages and in the final stage separately. Tables 6.3, 6.4 and 6.5 show the results when BFP procedure is applied only in the final stage. In these tables, the average values of the results of the instances belonging the same instance subgroup are presented as ACMAX, ANPV and ANNS. In addition, the percent improvements with respect to Cmax and Npv are presented. The columns under the heading Without BFP on the Archive correspond to pure application of NSGA-II to the test problems. The columns under the heading BFP on the Archive correspond, on the other hand, to ACMAX, ANPV, and ANNS obtained after BFP is applied to the archive of non-dominated solutions resulting from the application of NSGA-II.

Table 6.6, 6.7 and 6.8 show the statistical results. In order to evaluate whether a significant difference between average Cmax values, average Npv values and average number of non-dominated solutions, paired t-test is applied. Since the same sample (instances) are used, paired t-test should be applied. In addition, since we are concerned about the existence of the equality between the means of average Cmax, average Npv and average number of non-dominated solutions, p value of two-tail result must be evaluated.

In order to provide better understanding of the results, the following figures may help. Figures 6.1, 6.2, 6.3, and 6.4 show the difference between the solutions obtained with pure NSGA-II application and the solutions obtained after BFP application on the archive of non-dominated solutions resulting from the application of NSGA-II. Recall that objectives are minCMAX and maxNPV. Thus, it is observed that BFP is significantly effective in getting better solutions in terms of objective values.

Abbreviations	Explanations
INSSUB	Instance Subgroups
INS	Instance
ACMAX	Average Cmax
ANPV	Average Npv
AMFT	Average of Mean Flow Times
AMWT	Average of Mean Weighted Tardiness
AMCT	Average of Mean Completion Time
ARUD	Average Resource Usage Deviation
AMO	Average of Minimum Outflows
ANNS	Average Number of Non-dominated Solutions
ACPU	Average CPU Times
% Imp.	Percent Improvement
H. Mean Difference	Hypothesized Mean Difference
D	Represents the Difficult Case of Difficulty
M	Represents the Medium Case of Difficulty
E	Represents the Easy Case of Difficulty
CPU(D)	CPU Times for solving Difficult Instances
CPU(M)	CPU Times for solving Medium Instances
CPU(E)	CPU Times for solving Easy Instances

Table 6.2: Abbreviations Used in Tables

INSSUB	Without BFP on the Archive by GA			BFP on the Archive				
	ACMAX	ANPV	ANNS	ACMAX	% Imp.	ANPV	% Imp.	ANNS
A11	61.39	8452.00	4.33	43.72	28.78	27448.15	224.75	1.33
A12	39.50	30205.81	2.22	29.67	24.89	65177.23	115.78	1.56
A13	38.33	35943.40	1.89	28.00	26.95	77852.86	116.60	1.00
A21	106.09	-3786.42	19.56	89.70	15.45	1313.29	134.68	13.67
A22	41.76	42472.89	1.67	31.50	24.57	91857.04	116.27	1.22
A23	39.06	46336.24	2.00	30.83	21.07	94562.36	104.08	1.11
A31	168.01	-9566.67	27.33	161.28	4.01	-9956.69	-4.08	28.78
A32	45.52	31616.04	1.44	35.56	21.88	82375.22	160.55	1.00
A33	43.50	54445.20	2.22	31.70	27.13	125864.36	131.18	1.56

Table 6.3: Results for Set A - BFP in the Final Stage

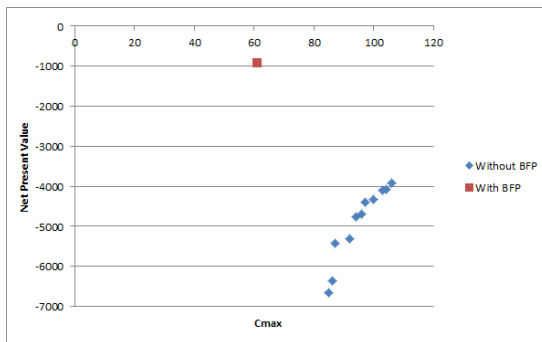


Figure 6.1: Instance A11\_11

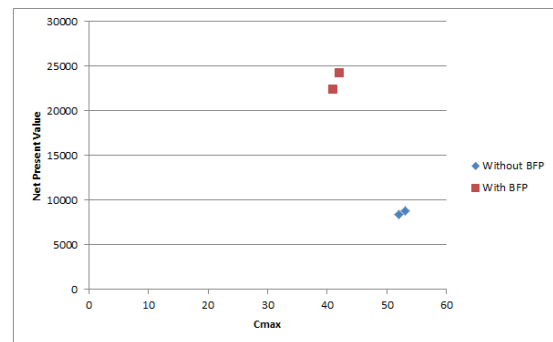


Figure 6.2: Instance A11\_21



INSSUB	Without BFP on the Archive by GA			BFP on the Archive				
	ACMAX	ANPV	ANNS	ACMAX	% Imp.	ANPV	% Imp.	ANNS
B1010	65.80	-4284.41	21.00	51.73	21.38	-2394.56	44.11	16.25
B1012	70.57	-3325.96	22.50	55.12	21.89	-1562.10	53.03	17.75
B1014	63.50	4193.34	4.25	46.63	26.57	15971.34	280.87	1.25
B1016	58.63	6312.65	2.25	39.25	33.05	36364.15	476.05	1.50
B1018	61.84	2653.08	3.00	45.00	27.23	31971.18	1105.06	1.50
B1020	60.38	1082.86	1.75	41.25	31.68	31269.59	2787.69	1.00
B1030	133.84	-9476.70	39.50	90.85	32.12	11441.63	220.73	16.25
B1510	62.82	9352.66	4.75	47.00	25.18	23303.23	149.16	1.75
B1512	51.75	12917.71	1.75	38.50	25.60	35089.88	171.64	1.00
B1514	79.17	-284.76	12.25	49.25	37.79	20654.11	7353.16	1.50
B1516	67.65	1117.90	4.00	43.38	35.88	39419.66	3426.22	1.50
B1518	85.54	-5074.28	15.25	46.88	45.20	30145.27	694.08	1.50
B1520	91.51	-6035.83	20.00	44.38	51.50	34819.41	676.88	1.75
B1530	169.76	-12662.37	55.00	77.13	54.57	13885.18	209.66	11.00
B2010	42.69	30931.28	2.25	33.38	21.81	65006.28	110.16	1.75
B2012	54.60	17293.89	2.25	41.75	23.53	47993.94	177.52	1.50
B2014	74.85	1809.18	9.25	47.13	37.03	33407.60	1746.56	1.50
B2016	73.38	-3739.17	6.25	43.00	41.40	62090.64	1760.55	1.25
B2018	119.10	-8377.08	35.25	51.13	57.07	28389.37	438.89	1.50
B2020	126.34	-10501.11	40.00	46.88	62.89	47820.26	555.38	2.50
B2030	201.89	-14610.14	73.50	83.88	58.45	14614.88	200.03	12.25

Table 6.4: Results for Set B - BFP in the Final Stage

INSSUB	Without BFP on the Archive by GA			BFP on the Archive				
	ACMAX	ANPV	ANNS	ACMAX	% Imp.	ANPV	% Imp.	ANNS
C1	104.38	-7449.74	18.78	54.07	48.20	38144.69	612.03	2.33
C2	72.72	12308.41	5.67	50.04	31.19	50549.32	310.69	2.22
C3	82.55	-3299.87	12.22	45.28	45.15	43385.62	1414.77	1.67

Table 6.5: Results for Set C - BFP in the Final Stage

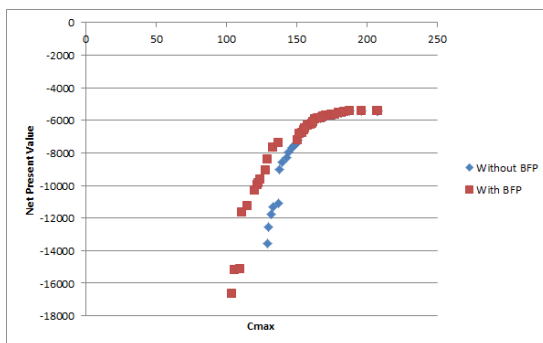


Figure 6.3: Instance A21\_11

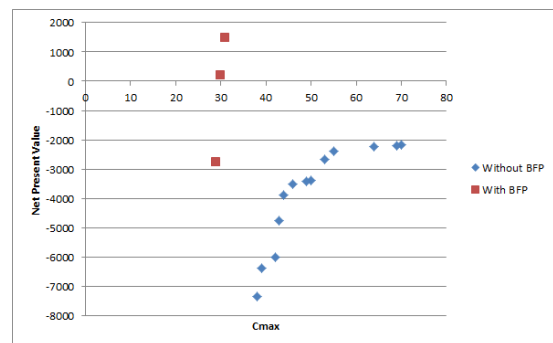


Figure 6.4: Instance B1010\_21

Statistics	<i>ACMAX</i> <sub>1</sub>	<i>ACMAX</i> <sub>2</sub>	<i>ANPV</i> <sub>1</sub>	<i>ANPV</i> <sub>2</sub>	<i>ANNS</i> <sub>1</sub>	<i>ANNS</i> <sub>2</sub>
Mean	64.80	53.55	26235.39	61832.65	6.96	5.69
Variance	1966.04	2004.47	512725072.6	2095148054	91.79	91.86
Observations	9	9	9	9	9	9
Pearson Correlation	0.99		0.99		0.98	
H. Mean Difference	0		0		0	
Degrees of Freedom	8		8		8	
t Stat	9.40		-4.55		1.84	
P(T ≤ t) one-tail	6.71535E-06		0.000935077		0.051160589	
t Critical one-tail	1.859548038		1.859548038		1.859548038	
P(T ≤ t) two-tail	1.34307E-05		0.001870153		0.102321177	
t Critical two-tail	2.306004135		2.306004135		2.306004135	

The column headings whose index is 1 present the results of Without BFP on the Archive  
The column headings whose index is 2 present the results of BFP on the Archive  
Significance level: 0.05

Table 6.6: Paired t-test Result for Set A - BFP in the Final Stage

Statistics	<i>ACMAX</i> <sub>1</sub>	<i>ACMAX</i> <sub>2</sub>	<i>ANPV</i> <sub>1</sub>	<i>ANPV</i> <sub>2</sub>	<i>ANNS</i> <sub>1</sub>	<i>ANNS</i> <sub>2</sub>
Mean	86.46	50.64	442.51	29509.57	17.90	4.65
Variance	1693.23	222.06	116296330.10	315250177.40	401.97	34.89
Observations	21	21	21	21	21	21
Pearson Correlation	0.84		0.51		0.60	
H. Mean Difference	0		0		0	
Degrees of Freedom	20		20		20	
t Stat	5.52		-8.63		3.54	
P(T ≤ t) one-tail	1.05E-05		1.76729E-08		0.001027384	
t Critical one-tail	1.724718243		1.724718243		1.724718243	
P(T ≤ t) two-tail	2.09571E-05		3.53458E-08		0.002054769	
t Critical two-tail	2.085963447		2.085963447		2.085963447	

The column headings whose index is 1 present the results of Without BFP on the Archive  
The column headings whose index is 2 present the results of BFP on the Archive  
Significance level: 0.05

Table 6.7: Paired t-test Result for Set B - BFP in the Final Stage

Statistics	$ACMAX_1$	$ACMAX_2$	$ANPV_1$	$ANPV_2$	$ANNS_1$	$ANNS_2$
Mean	86.55	49.80	519.60	44026.54	12.22	2.07
Variance	262.59	19.36	108537386.20	38776798.40	42.97	0.13
Observations	3	3	3	3	3	3
Pearson Correlation	0.6		0.97		0.16	
H. Mean Difference	0		0		0	
Degrees of Freedom	2		2		2	
t Stat	4.61		-16.41		2.70	
P(T ≤ t) one-tail	0.022026853		0.001847219		0.057050024	
t Critical one-tail	2.91998558		2.91998558		2.91998558	
P(T ≤ t) two-tail	0.044053707		0.003694438		0.114100048	
t Critical two-tail	4.30265273		4.30265273		4.30265273	

The column headings whose index is 1 present the results of Without BFP on the Archive

The column headings whose index is 2 present the results of BFP on the Archive

Significance level: 0.05

Table 6.8: Paired t-test Result for Set C - BFP in the Final Stage

A similar analysis is performed, but this time under BFP in the intermediate stages by considering the same objective combinations, minCMAX/maxNPV. Tables 6.9, 6.10 and 6.11 present the ACMAX, ANPV and ANNS under different scenarios. In the first scenario under the heading GA Application, these quantities are obtained with pure application of NSGA-II. It is noted that these values are the same with the values under the heading Without BFP on the Archive by GA in Tables 6.3, 6.4 and 6.5. The second scenario becomes applying BFP in the intermediate stages, but without applying BFP on the archive. The last scenario is to apply both, BFP in the intermediate stages and BFP on the archive.

It may be beneficial to show how frequently BFP is applied during the implementation of GA in order to understand the behavior of the algorithm. Table 6.12 shows these results. The second, sixth and tenth columns denote the bounds of generation numbers, which are determined before implementation of GA. The values in the next columns show the average numbers of BFP operation and the average last generations of the algorithm. For example, BFP is applied on the average 4.44 times during 275.11 average number of generations in A11 subgroup of set A.

In addition to BFP observation, the efficiency of the entropy-based divergence application is evaluated as well to assign the best threshold values. Similar to BFP, the behavior of the algorithm with respect to activity entropy and mode entropy is observed. In other words, it is studied how strictly activity and mode entropy values change from one generation to the next generation. For this purpose, the generation numbers in which the activity

INSSUB	GA Application			BFP in the Intermediate Stages					
				Without BFP on the Archive			BFP on the Archive		
	ACMAX	ANPV	ANNS	ACMAX	ANPV	ANNS	ACMAX	ANPV	ANNS
A11	61.39	8452.00	4.33	50.16	26130.72	3.78	43.39	27585.80	2.11
A12	39.50	30205.81	2.22	29.05	68494.32	2.44	29.05	68494.32	2.44
A13	38.33	35943.40	1.89	28.30	77474.89	2.44	28.30	77474.89	2.44
A21	106.09	-3786.42	19.56	89.65	-694.03	12.56	88.49	-773.11	12.33
A22	41.76	42472.89	1.67	29.55	97170.35	2.89	29.32	97863.80	2.67
A23	39.06	46336.24	2.00	29.66	100150.57	2.67	29.41	100415.73	2.33
A31	168.01	-9566.67	27.33	158.28	-9101.59	26.33	155.10	-9316.50	27.78
A32	45.52	31616.04	1.44	35.81	85830.11	2.11	35.81	85884.65	2.11
A33	43.50	54445.20	2.22	29.02	135491.89	2.33	29.06	135579.90	2.44

Table 6.9: Results for Set A - BFP in the Intermediate Stages

INSSUB	GA Application			BFP in the Intermediate Stages					
				Without BFP on the Archive			BFP on the Archive		
	ACMAX	ANPV	ANNS	ACMAX	ANPV	ANNS	ACMAX	ANPV	ANNS
B1010	65.80	-4284.41	21.00	62.59	-2450.04	21.25	55.18	-1645.53	14.50
B1012	70.57	-3325.96	22.50	60.90	-2605.09	19.75	51.57	-1693.82	15.75
B1014	63.50	4193.34	4.25	45.88	18307.21	2.00	45.81	18297.54	2.25
B1016	58.63	6312.65	2.25	38.25	37948.59	2.25	38.25	37948.59	2.25
B1018	61.84	2653.08	3.00	51.08	26680.20	3.75	42.50	31210.53	1.75
B1020	60.38	1082.86	1.75	40.19	36900.63	3.00	40.19	36900.63	3.00
B1030	133.84	-9476.70	39.50	76.92	11983.28	5.25	76.92	11983.28	5.25
B1510	62.82	9352.66	4.75	52.67	22824.77	3.50	46.67	25595.70	1.75
B1512	51.75	12917.71	1.75	38.96	37206.15	1.75	38.96	37206.15	1.75
B1514	79.17	-284.76	12.25	50.13	20227.01	2.25	50.13	20227.01	2.25
B1516	67.65	1117.90	4.00	53.72	35707.76	3.75	43.00	41429.02	1.75
B1518	85.54	-5074.28	15.25	59.42	28556.16	8.75	46.33	32060.41	2.00
B1520	91.51	-6035.83	20.00	44.88	33673.82	2.00	44.88	33673.82	2.00
B1530	169.76	-12662.37	55.00	85.66	14748.95	22.00	85.40	14705.95	22.25
B2010	42.69	30931.28	2.25	31.88	64775.01	2.25	31.88	64775.01	2.25
B2012	54.60	17293.89	2.25	40.75	49456.40	3.00	40.75	49456.40	3.00
B2014	74.85	1809.18	9.25	58.33	33136.85	8.25	47.38	37069.28	1.75
B2016	73.38	-3739.17	6.25	41.81	64622.47	2.00	41.81	64622.47	2.00
B2018	119.10	-8377.08	35.25	50.00	28125.86	2.50	50.00	28125.86	2.50
B2020	126.34	-10501.11	40.00	47.49	50985.14	2.75	47.49	50985.14	2.75
B2030	201.89	-14610.14	73.50	90.22	17661.04	27.25	89.82	17578.70	27.75

Table 6.10: Results for Set B - BFP in the Intermediate Stages

INSSUB	GA Application			BFP in the Intermediate Stages					
				Without BFP on the Archive			BFP on the Archive		
	ACMAX	ANPV	ANNS	ACMAX	ANPV	ANNS	ACMAX	ANPV	ANNS
C1	104.38	-7449.74	18.78	60.15	38602.39	6.11	59.62	38812.37	5.44
C2	72.72	12308.41	5.67	49.53	55377.17	3.56	49.53	55380.66	3.56
C3	82.55	-3299.87	12.22	49.92	45725.56	5.89	44.31	46359.41	1.89

Table 6.11: Results for Set C - BFP in the Intermediate Stages

INSSUB	Bounds on Generation Numbers	Average Number of BFP Operations	Last Generation	INSSUB	Bounds on Generation Numbers	Average Number of BFP Operations	Last Generation	INSSUB	Bounds on Generation Numbers	Average Number of BFP Operations	Last Generation
A11	350	4.44	275.11	B1014	350	5.50	187.00	B1530	1125	7.50	663.75
A12	350	6.56	332.44	B1016	400	6.75	313.00	B2010	500	7.50	296.00
A13	350	6.78	336.67	B1018	450	5.25	368.25	B2012	600	6.00	430.00
A21	350	6.11	265.89	B1020	500	6.25	403.75	B2014	700	4.75	473.75
A22	350	6.11	300.33	B1030	750	8.00	614.25	B2016	800	7.25	579.25
A23	350	6.22	303.56	B1510	375	4.50	306.25	B2018	900	9.00	520.75
A31	350	3.33	338.44	B1512	450	5.75	393.50	B2020	1000	7.00	599.00
A32	350	8.33	297.44	B1514	525	7.50	245.50	B2030	1500	6.25	897.75
A33	350	7.11	327.00	B1516	600	5.00	453.75	C1	750	5.11	517.33
B1010	250	1.50	241.00	B1518	675	5.00	500.00	C2	590	6.67	388.22
B1012	300	2.50	268.75	B1520	750	5.50	428.25	C3	740	4.78	547.78

Table 6.12: Frequency of BFP During the Implementation of GA

and mode entropy measure values reach to  $[1, 0.98, 0.96, \dots, 0.22]$  are recorded. By using the same classification of the instances in BFP, the average values of the recorded generation numbers in the same subgroups of the instances are calculated. The first entropy value whose average generation number is larger than half of the bound of generation number is labeled as the threshold for activity and mode entropy measures. This procedure is performed for activity and mode entropy separately.

The scale of this experiment is kept small and only set A is considered. The threshold values of the activity and mode entropy measures of the instance subgroups are shown in Table 6.13. The first and second divisions show the different levels for the threshold values because both are intended to be experimented.

It should be noted that another important parameter in this experiment is how many individuals are removed from the population. It is rational that this number is a proportion of the population size. Thus, two different proportions, which are 0.25 and 0.15 are used in this experiment. Finally, the algorithm is performed for four combinations of first and second level thresholds and the proportions of population size which will be removed. Tables 6.14 and 6.15 show the results of this experiment. In these tables,  $R = 0.25$  and  $R = 0.15$  denote that 25 percent and 15 percent of the population size are removed from the population, respectively.

Tables 6.16 and 6.17 show how frequently entropy-based divergence application is

INSSUB	First Level Thresholds For		Second Level Thresholds For	
	Activity Entropy	Mode Entropy	Activity Entropy	Mode Entropy
A11	0.42	0.78	0.44	0.80
A12	0.40	0.78	0.42	0.80
A13	0.42	0.76	0.44	0.78
A21	0.42	0.76	0.44	0.78
A22	0.38	0.76	0.40	0.78
A23	0.40	0.76	0.43	0.78
A31	0.44	0.72	0.46	0.74
A32	0.40	0.78	0.42	0.80
A33	0.40	0.76	0.42	0.78

Table 6.13: Activity and Mode Entropy Thresholds

INSSUB	First Level & R=0.25				First Level & R=0.15			
	ACMAX	ANPV	ANNS	ACPU	ACMAX	ANPV	ANNS	ACPU
A11	65.15	8182.67	5.22	150.30	63.83	8264.17	5.00	148.46
A12	36.50	34437.61	1.22	140.94	39.54	28571.77	1.67	141.97
A13	38.83	34835.26	1.67	144.39	38.28	36192.52	1.89	143.34
A21	106.40	-4627.70	17.78	156.56	106.02	-4219.21	17.78	155.97
A22	43.19	39214.96	2.33	144.29	42.89	37918.21	1.67	143.69
A23	41.07	46453.15	2.11	144.14	41.78	44196.59	2.00	144.04
A31	168.79	-9662.36	28.00	165.08	173.22	-9933.66	29.11	164.39
A32	45.83	28138.41	1.89	146.43	44.94	33148.12	1.56	144.48
A33	44.27	50733.86	2.22	145.85	41.19	56085.39	1.67	142.70

Table 6.14: Results for Entropy-Based Divergence - First Level

INSSUB	Second Level & R=0.25				Second Level & R=0.15			
	ACMAX	ANPV	ANNS	ACPU	ACMAX	ANPV	ANNS	ACPU
A11	64.83	6497.35	5.89	150.43	65.22	7585.52	5.22	150.65
A12	38.61	32341.76	2.33	142.86	37.56	31887.38	1.89	142.17
A13	39.20	32794.35	1.44	145.68	38.39	34714.21	1.78	144.49
A21	107.52	-4591.78	19.33	156.93	107.74	-4589.25	19.56	156.91
A22	43.59	40411.98	2.22	144.25	43.50	36124.90	2.00	143.74
A23	40.52	45318.95	1.78	143.37	40.20	43606.30	2.33	143.84
A31	171.16	-10143.21	27.00	165.88	173.49	-10009.68	28.78	165.31
A32	47.09	29067.71	2.22	146.45	44.44	31116.75	1.89	145.82
A33	40.56	57032.05	1.78	143.15	40.72	58656.40	2.00	143.82

Table 6.15: Results for Entropy-Based Divergence - Second Level

INSSUB	First Level & R=0.25		First Level & R=0.15	
	Activity	Mode	Activity	Mode
A11	4.56	5.33	5.89	6.22
A12	1.00	7.22	3.56	7.78
A13	5.56	3.56	7.33	6.89
A21	0.33	4.22	3.56	6.67
A22	0.22	4.00	0.33	6.89
A23	1.22	4.33	2.44	7.56
A31	2.67	1.78	2.33	1.78
A32	1.89	7.00	3.22	9.44
A33	4.33	3.33	4.89	7.22

Table 6.16: Frequency for Entropy-Based Divergence - First Level

INSSUB	Second Level & R=0.25		Second Level & R=0.15	
	Activity	Mode	Activity	Mode
A11	5.78	6.89	19.44	9.22
A12	6.44	8.00	6.11	12.11
A13	11.44	4.78	17.22	5.56
A21	2.78	6.44	6.89	9.33
A22	3.00	6.89	3.56	9.89
A23	3.56	6.33	7.56	8.67
A31	5.33	2.67	5.11	4.11
A32	5.78	9.67	7.00	13.33
A33	6.11	4.33	8.00	9.22

Table 6.17: Frequency for Entropy-Based Divergence - Second Level

operated during the implementation of the algorithm. For example, when the first level threshold values are used and 25 percent of the population size are removed in the case of application, the activity entropy measure of the algorithm decreases under the corresponding threshold value 4,56 times in average for A11 subgroup of set A. It should be recalled that generation number for the set A is 350.

### 6.1.2 Implementation with the test instances generated in this thesis

As for test instances generated in this thesis, they are solved with GA by experimenting different difficulties with respect to due date, resource capacities and lump sum payments. Since the objective combination minCMAX/maxNPV has been studied before, the test instances generated in this thesis are performed with the other objective combinations. The purpose of solving these instances is to observe the behavior of the algorithm while performing it with different due date difficulties, resource capacity difficulties and lump

sum payment difficulties. In addition to this goal, it is desired to be observed whether BFP is efficient with the objective combinations mentioned above. However, BFP is applied only in the intermediate stages because it is observed in the previous experiments that it is more efficient than BFP in the final stage.

The instances selected for this experiment are listed as below:

- Instance Subgroup 1: 10 Projects, 10 Activities, 9 Instances (3 instances for different due date difficulties, 3 instances for different lump sum payment difficulties, 3 instances for different resource capacity difficulties)
- Instance Subgroup 2: 10 Projects, 12 Activities, 9 Instances (3 instances for different due date difficulties, 3 instances for different lump sum payment difficulties, 3 instances for different resource capacity difficulties)
- Instance Subgroup 3: 15 Projects, 10 Activities, 9 Instances (3 instances for different due date difficulties, 3 instances for different lump sum payment difficulties, 3 instances for different resource capacity difficulties)
- Instance Subgroup 4: 15 Projects, 12 Activities, 9 Instances (3 instances for different due date difficulties, 3 instances for different lump sum payment difficulties, 3 instances for different resource capacity difficulties)
- Instance Subgroup 5: 20 Projects, 10 Activities, 9 Instances (3 instances for different due date difficulties, 3 instances for different lump sum payment difficulties, 3 instances for different resource capacity difficulties)
- Instance Subgroup 6: 20 Projects, 12 Activities, 9 Instances (3 instances for different due date difficulties, 3 instances for different lump sum payment difficulties, 3 instances for different resource capacity difficulties)

That is, total number of activities in instance subgroups are 100, 120, 150, 180, 200 and 240, respectively. Additionally, total number of instances is 54.

Firstly, these instances are solved with the objective combination maxNPV/minMFT. Tables 6.18 shows the average Npv and mean flow time values of the instances. Moreover, Tables 6.19 shows the corresponding CPU times in seconds.

Secondly, these instances are solved with the objective combination maxNPV/minMWT. Table 6.20 shows the average Npv and mean weighted tardiness values of the instances. Moreover, Table 6.21 shows the corresponding CPU times in seconds.



INS	Difficult			Medium			Easy		
	ANPV	AMFT	ANNS	ANPV	AMFT	ANNS	ANPV	AMFT	ANNS
Due Date Difficulty									
1	2314.78	32.77	3	2380.85	31.83	4	2672.70	30.93	3
2	1123.85	37.80	1	1247.38	38.85	2	731.30	40.00	1
3	1269.44	37.52	4	1200.33	40.22	4	891.57	40.18	4
4	605.11	43.27	5	807.07	42.20	1	473.85	42.19	5
5	-162.28	42.68	3	738.06	42.83	4	989.34	40.45	1
6	-503.92	47.68	5	116.55	42.33	5	-629.62	50.92	3
Lump Sum Payment Difficulty									
1	195.88	32.60	2	2303.15	34.80	1	5832.06	30.70	1
2	-696.67	41.80	4	1008.54	39.08	4	2926.05	35.77	3
3	-1397.47	37.24	3	1002.51	38.42	7	5813.80	37.27	4
4	-1797.70	55.40	15	-149.13	46.47	2	3140.24	42.11	3
5	-2499.37	45.60	4	-185.17	43.81	7	4622.97	41.12	3
6	-2137.03	53.33	8	-838.97	47.03	6	2497.96	43.30	2
Resource Capacity Difficulty									
1	2720.48	31.60	1	3510.08	28.40	3	3600.68	26.67	3
2	939.93	38.18	5	1994.76	30.77	3	1805.46	31.98	4
3	1419.42	39.67	2	2924.25	33.16	5	1916.94	38.80	4
4	397.20	43.00	2	1306.38	38.36	3	1446.32	38.44	5
5	-57.01	43.30	3	1443.12	38.62	3	2116.93	37.43	3
6	-272.40	46.29	8	1261.27	40.90	2	352.67	41.67	6

Table 6.18: Results for maxNPV/minMFT

INS	Due Date			Lump Sum Payment			Resource Capacity		
	CPU(D)	CPU(M)	CPU(E)	CPU(D)	CPU(M)	CPU(E)	CPU(D)	CPU(M)	CPU(E)
1	46.40	46.82	46.49	47.53	46.80	45.45	46.24	46.86	47.56
2	95.99	94.65	97.28	97.15	94.73	92.04	94.76	96.34	96.91
3	217.06	216.98	216.24	218.28	217.73	211.77	214.35	215.99	217.68
4	451.78	445.99	451.05	458.60	456.85	447.24	454.63	459.38	442.08
5	655.83	656.79	643.88	657.79	658.83	650.16	655.46	650.33	648.38
6	1353.11	1353.42	1364.62	1379.73	1372.18	1344.08	1364.38	1329.67	1345.29

Population Size: 126, 150, 188, 226, 250, 300, respectively  
Generation Number: 300, 360, 450, 540, 600, 720, respectively

Table 6.19: CPU Times for maxNPV/minMFT

INS	Difficult			Medium			Easy		
	ANPV	AMWT	ANNS	ANPV	AMWT	ANNS	ANPV	AMWT	ANNS
Due Date Difficulty									
1	2602.86	0.00	1	2843.05	0.00	1	2666.27	0.00	1
2	1321.09	0.00	1	1111.23	0.00	1	500.60	0.00	1
3	1980.81	0.07	2	805.66	0.03	2	1381.30	0.00	1
4	187.58	2.30	6	-142.53	1.00	2	-764.14	1.68	5
5	316.43	0.48	2	-1803.55	7.46	8	288.15	0.08	2
6	-737.08	1.23	8	96.56	0.00	1	-820.20	0.76	4
Lump Sum Payment Difficulty									
1	172.13	0.00	1	2406.37	0.05	2	4672.87	0.00	1
2	-876.88	4.48	6	1345.44	0.00	1	3228.14	0.00	1
3	-1521.80	1.43	5	1734.01	0.03	2	4469.10	0.00	1
4	-1586.60	10.40	17	-212.19	0.58	3	3139.89	0.00	1
5	-3232.60	5.94	7	548.81	0.08	2	6493.79	0.00	1
6	-2568.69	10.86	19	-21.53	1.78	3	3393.50	0.00	1
Resource Capacity Difficulty									
1	2373.63	0.00	1	3780.60	0.00	1	3616.15	0.00	1
2	1312.20	0.00	1	2074.74	0.00	1	2003.49	0.00	1
3	1095.13	0.10	2	1843.73	0.00	1	3562.30	0.00	1
4	-13.78	0.11	3	1543.94	0.00	1	1874.56	0.00	1
5	840.21	0.10	2	1561.49	0.00	1	1460.31	0.15	2
6	162.90	0.05	2	999.55	0.00	1	-427.63	0.27	3

Table 6.20: Results for maxNPV/minMWT

INS	Due Date			Lump Sum Payment			Resource Capacity		
	CPU(D)	CPU(M)	CPU(E)	CPU(D)	CPU(M)	CPU(E)	CPU(D)	CPU(M)	CPU(E)
1	29.96	29.85	29.91	30.62	29.63	30.29	30.21	30.83	30.67
2	61.71	63.18	62.41	63.72	61.67	61.95	61.54	62.55	62.61
3	138.99	139.44	142.14	143.22	138.47	139.74	141.60	142.59	140.20
4	296.51	295.91	297.59	300.56	295.11	290.26	293.27	295.05	293.11
5	419.76	431.77	422.26	431.40	424.69	412.92	425.40	431.74	428.88
6	893.51	890.36	897.20	905.16	894.98	860.06	897.78	891.15	898.15

Population Size: 100, 120, 150, 180, 200, 240, respectively  
Generation Number: 250, 300, 375, 450, 500, 600, respectively

Table 6.21: CPU Times for maxNPV/minMWT

INS	Difficult			Medium			Easy		
	ANPV	AMCT	ANNS	ANPV	AMCT	ANNS	ANPV	AMCT	ANNS
Due Date Difficulty									
1	2330.98	34.70	2	2919.73	32.00	1	2593.29	33.73	3
2	881.41	41.23	3	917.89	40.80	2	808.96	42.90	3
3	2399.85	39.31	3	2192.51	37.73	2	1410.81	42.87	3
4	403.59	43.75	4	989.16	41.37	2	84.09	46.30	6
5	2114.39	39.93	5	782.43	45.35	5	1036.96	42.07	3
6	183.18	44.85	6	260.57	43.45	5	-803.83	61.92	10
Lump Sum Payment Difficulty									
1	-431.44	34.55	2	3244.05	30.40	1	5936.35	30.80	1
2	-912.83	49.49	8	1528.43	35.30	2	2363.57	41.57	3
3	-1974.80	47.12	4	2202.94	36.84	3	6407.08	36.30	2
4	-1524.79	61.31	10	168.99	44.13	3	3065.23	42.40	4
5	-3142.01	58.55	8	-193.93	47.25	4	4718.38	43.60	4
6	-2755.08	75.26	31	-433.98	50.00	6	2977.30	44.92	3
Resource Capacity Difficulty									
1	2753.10	33.68	4	3698.37	28.80	1	3928.21	26.75	2
2	1094.99	37.35	2	2133.45	31.13	3	1932.18	31.94	5
3	1991.18	39.53	2	2805.99	37.12	4	3699.62	31.91	3
4	436.31	45.67	2	1355.08	37.03	2	1507.55	37.79	6
5	405.08	51.04	7	3418.17	36.02	3	2226.67	37.80	5
6	95.50	45.72	5	1775.36	39.35	1	-373.76	49.29	4

Table 6.22: Results for maxNPV/minMCT

In addition to these objectives, the instances are solved with the objective combinations maxNPV/minMCT. Table 6.22 shows the average Npv and mean completion time values of the instances. Moreover, Table 6.23 shows the corresponding CPU times in seconds.

As for the next experiment, these instances are solved with the objective combination minCMAX/maxNPV/minRUD. Table 6.24 shows the average Cmax, average Npv and average resource usage deviations of the instances. Moreover, Table 6.25 shows the corresponding CPU times in seconds.

Finally, these instances are solved with the objective combination minCMAX/maxNPV/maxMO. Tables 6.26 shows the average Cmax, average Npv and minimum outflow of the instances. Moreover, Table 6.27 shows the corresponding CPU times in seconds.

In order to observe whether BFP is efficient for these objective combinations, some experiments are performed. While solving the instances at first, discontinuation of the

INS	Due Date			Lump Sum Payment			Resource Capacity		
	CPU(D)	CPU(M)	CPU(E)	CPU(D)	CPU(M)	CPU(E)	CPU(D)	CPU(M)	CPU(E)
1	46.42	45.71	46.14	46.20	45.76	44.76	46.52	47.38	46.88
2	93.34	94.96	93.45	98.13	92.18	92.88	92.73	93.54	93.82
3	209.27	210.08	216.13	216.92	208.39	206.75	213.18	211.75	210.39
4	436.79	432.35	442.65	449.65	427.14	426.60	447.89	435.32	432.14
5	633.85	645.08	635.30	654.34	642.10	625.69	641.18	632.97	626.98
6	1297.86	1299.65	1363.93	1375.28	1346.56	1312.06	1337.46	1303.21	1343.46

Population Size: 126, 150, 188, 226, 250, 300, respectively  
Generation Number: 300, 360, 450, 540, 600, 720, respectively

Table 6.23: CPU Times for maxNPV/minMCT

INS	Difficult				Medium				Easy			
	ACMAX	ANPV	ARUD	ANNS	ACMAX	ANPV	ARUD	ANNS	ACMAX	ANPV	ARUD	ANNS
Due Date Difficulty												
1	99.96	-2172.56	7.97	51	110.15	-2068.12	8.59	100	81.24	-1659.50	9.81	54
2	107.43	-1541.18	6.07	77	124.01	-1565.33	5.88	135	109.94	-1458.04	6.41	145
3	119.64	-3464.77	12.45	158	143.52	-3934.69	11.76	221	134.19	-3944.01	12.13	245
4	149.34	-2469.98	7.94	377	176.02	-2573.07	7.26	460	144.24	-2298.65	8.09	322
5	168.28	-5057.99	14.69	588	161.55	-4700.64	16.63	778	181.87	-4548.20	14.88	796
6	187.91	-3061.22	10.39	1115	169.22	-3328.57	9.84	728	186.37	-3075.58	10.45	974
Lump Sum Payment Difficulty												
1	106.80	-2687.54	9.09	133	72.27	-1254.62	10.99	62	78.28	-469.23	10.13	32
2	116.08	-2004.31	6.12	166	113.17	-1547.40	5.50	54	91.76	-492.10	6.01	45
3	140.47	-4208.61	13.35	435	145.46	-3528.92	10.75	199	148.56	-3226.39	9.79	96
4	144.85	-3070.91	8.90	563	167.53	-2219.48	8.26	573	168.15	-1897.28	7.27	312
5	159.79	-5350.26	16.48	716	165.34	-4854.95	14.25	560	170.23	-4323.37	14.07	416
6	175.84	-3804.74	10.51	1074	186.81	-3204.16	10.24	940	188.57	-2947.06	9.34	679
Resource Capacity Difficulty												
1	73.33	-1051.78	10.87	63	77.64	-339.75	10.18	56	65.59	-198.02	11.11	39
2	131.56	-1459.22	5.95	126	85.66	-826.55	6.67	67	115.51	-1444.93	6.04	116
3	143.58	-3327.58	12.35	342	145.51	-3929.64	10.14	164	144.73	-3094.60	11.03	216
4	135.72	-2371.84	8.10	240	148.61	-2505.77	7.04	199	164.28	-2082.87	7.41	475
5	162.46	-4568.76	16.16	653	176.60	-4556.24	13.72	601	185.35	-4535.24	13.27	653
6	182.57	-3399.02	9.62	900	196.82	-3089.85	9.13	809	184.41	-3080.62	8.80	654

Table 6.24: Results for minCMAX/maxNPV/minRUD

INS	Due Date			Lump Sum Payment			Resource Capacity		
	CPU(D)	CPU(M)	CPU(E)	CPU(D)	CPU(M)	CPU(E)	CPU(D)	CPU(M)	CPU(E)
1	49.58	50.23	48.99	50.11	48.40	49.02	49.12	50.89	51.05
2	103.52	104.89	103.51	104.13	102.97	102.71	104.94	102.70	105.83
3	229.03	237.79	235.44	238.15	234.71	234.13	237.48	238.51	240.10
4	481.18	487.97	482.13	479.92	487.04	493.94	478.55	486.65	486.95
5	694.24	691.04	703.34	695.25	695.82	696.78	689.25	709.38	710.63
6	1453.35	1427.51	1450.88	1449.41	1449.80	1439.01	1454.16	1473.15	1452.44

Population Size: 150, 180, 226, 270, 300, 360, respectively  
Generation Number: 250, 300, 375, 450, 500, 600, respectively

Table 6.25: CPU Times for minCMAX/maxNPV/minRUD

INS	Difficult				Medium				Easy			
	ACMAX	ANPV	AMO	ANNS	ACMAX	ANPV	AMO	ANNS	ACMAX	ANPV	AMO	ANNS
Due Date Difficulty												
1	72.69	745.64	-3866.55	35	79.19	70.93	-4719.04	63	74.35	375.03	-4134.90	34
2	83.08	-218.93	-3517.75	87	81.32	2.58	-3322.99	82	100.91	-413.94	-2963.66	74
3	97.40	-1190.84	-6422.78	94	101.45	-1293.86	-6214.90	93	92.86	-1057.42	-6498.05	86
4	101.47	-478.25	-4311.34	144	112.56	-840.43	-3966.47	118	109.72	-884.04	-4106.28	109
5	106.36	-1336.48	-7936.36	122	95.31	-964.69	-8120.73	119	100.95	-1040.84	-7016.12	102
6	115.68	-1153.64	-5025.49	109	121.31	-1387.14	-5161.37	93	105.39	-877.03	-5604.20	141
Lump Sum Payment Difficulty												
1	77.14	-1184.61	-4743.77	64	73.35	596.22	-3918.46	48	68.91	2404.94	-3895.79	35
2	95.34	-1029.00	-3181.42	94	78.25	138.78	-3244.73	60	83.95	634.11	-3444.09	77
3	93.03	-2627.97	-6587.96	65	90.09	-865.54	-6483.27	44	85.39	1693.97	-4826.69	49
4	109.78	-1544.41	-4137.29	130	100.09	-618.27	-4478.89	116	95.69	381.12	-4755.93	121
5	114.06	-2712.06	-7055.07	150	107.62	-1084.80	-7068.50	129	110.71	2.90	-7429.39	112
6	121.44	-2001.83	-4834.51	154	119.34	-989.80	-4781.64	132	102.56	339.90	-6420.26	161
Resource Capacity Difficulty												
1	80.22	43.39	-4440.91	64	82.56	357.59	-3749.93	75	73.89	605.01	-4010.43	66
2	81.98	-79.62	-3410.03	88	74.49	734.85	-2782.68	41	80.00	513.34	-2537.35	71
3	95.77	-966.23	-6348.02	94	90.09	-262.19	-5545.99	67	86.25	31.89	-6299.82	97
4	105.01	-803.41	-4510.44	165	92.56	81.31	-4395.31	125	98.09	-336.16	-4094.31	128
5	103.31	-904.35	-6896.77	120	101.50	-878.29	-7640.89	121	97.79	-834.90	-7824.30	130
6	115.96	-1177.44	-5190.22	171	108.03	-256.27	-4731.20	116	102.93	-698.48	-6024.81	156

Table 6.26: Results for minCMAX/maxNPV/maxMO

INS	Due Date			Lump Sum Payment			Resource Capacity		
	CPU(D)	CPU(M)	CPU(E)	CPU(D)	CPU(M)	CPU(E)	CPU(D)	CPU(M)	CPU(E)
1	48.60	49.33	49.16	48.82	49.01	48.73	49.31	51.35	51.16
2	101.70	101.31	102.25	101.48	101.42	101.22	101.79	102.53	102.29
3	224.90	227.18	227.61	227.16	225.45	225.29	226.78	229.60	231.99
4	466.82	469.98	466.20	466.82	464.07	464.28	469.04	463.80	465.14
5	670.83	670.38	672.64	676.25	671.46	675.07	670.54	680.69	672.10
6	1400.71	1396.96	1390.42	1404.23	1391.66	1397.81	1394.22	1390.59	1385.37

Population Size: 150, 180, 226, 270, 300, 360, respectively  
Generation Number: 250, 300, 375, 450, 500, 600, respectively

Table 6.27: CPU Times for minCMAX/maxNPV/maxMO

INS	maxNPV/minMFT		maxNPV/minMWT		maxNPV/minMCT	
	Threshold	Stoppage	Threshold	Stoppage	Threshold	Stoppage
1	13	5	15	5	13	5
2	15	5	15	5	15	5
3	17	5	15	5	21	5
4	19	5	13	5	15	5
5	17	5	15	5	17	5
6	15	5	15	5	19	5

Table 6.28: Threshold and Stoppage Values for Different Objective Combinations

archive is recorded. Similar to previous experiments, the same method for determining the threshold values for BFP in the intermediate stages are used. BFP in the final stage is not applied because it is observed not to be more efficient than BFP in the intermediate stages.

Firstly, the threshold values of the instances are given. These tables are prepared for each objective combination separately. Table 6.28 shows the threshold values for BFP in the intermediate stages. These values are not given for the objective combinations minC-MAX/maxNPV/minRUD and minC-MAX/maxNPV/maxMO because the discontinuation level can not reach to even 5. This means that the ability of the archive to renewing itself is excellent for these objective combinations. Thus, for the corresponding objective combinations, BFP is applied only in the final stage.

The results of the BFP in the intermediate stages are shown in the following tables. For the objective combination maxNPV/minMFT, Table 6.29 show the average Npv, mean flow time and the number of non-dominated solutions obtained with different scenarios, which are explained for Tables 6.9, 6.10 and 6.11. For the following tables, the letters D, M and E in the second columns corresponds to the different difficulty levels of the factors in the first columns. (D: difficult, M: medium, E: easy)

For the objective combination maxNPV/minMWT, the algorithm is performed with applying BFP in the intermediate stages by using the threshold values in Table 6.28. Table 6.30 shows the corresponding results.

For the objective combination maxNPV/minMCT, the algorithm is performed with applying BFP in the intermediate stages by using the threshold values in Table ???. Table 6.30 show the corresponding results.

As for the objective combinations minC-MAX/maxNPV/minRUD and minC-MAX/maxNPV/maxMO, BFP is applied only in the final stage. It should be noted that the number of solutions in instances whose number of activity is 240 is very high. Thus, during applying BFP on

INS		GA Application			BFP in the Intermediate Stages					
					Without BFP on the Archive			BFP on the Archive		
		ANPV	AMFT	ANNS	ANPV	AMFT	ANNS	ANPV	AMFT	ANNS
Due	D	774.50	40.29	3.50	3816.81	28.05	3.00	3797.25	28.01	3.17
Date	M	1081.71	39.71	3.33	3550.47	28.48	3.17	3603.50	28.47	2.83
Difficulty	E	854.86	40.78	2.83	3908.02	27.90	2.33	3908.02	27.90	2.33
Lump Sum	D	-1388.73	44.33	6.00	-616.84	37.48	7.00	-78.28	28.46	2.33
Payment	M	523.49	41.60	4.50	3544.67	28.66	2.50	3544.67	28.66	2.50
Difficulty	E	4138.85	38.38	2.67	7139.35	28.55	2.33	7139.35	28.55	2.33
Resource	D	857.94	40.34	3.50	3520.85	28.21	2.83	3520.85	28.21	2.83
Capacity	M	2073.31	35.03	3.17	4856.74	24.38	2.00	4856.74	24.38	2.00
Difficulty	E	1873.17	35.83	4.17	4983.56	23.71	2.33	4983.56	23.71	2.33

Table 6.29: The Result of BFP Procedure for maxNPV/minMFT

INS		GA Application			BFP in the Intermediate Stages					
					Without BFP on the Archive			BFP on the Archive		
		ANPV	AMWT	ANNS	ANPV	AMWT	ANNS	ANPV	AMWT	ANNS
Due	D	945.28	0.68	3.33	3946.53	0.00	1	3946.53	0.00	1
Date	M	485.07	1.41	2.50	3640.64	0.00	1	3651.42	0.00	1
Difficulty	E	542.00	0.42	2.33	3483.58	0.00	1	3483.58	0.00	1
Lump Sum	D	-1602.41	5.52	9.17	-365.25	1.21	3	226.05	0.00	1
Payment	M	966.82	0.42	2.17	3917.38	0.00	1	3917.38	0.00	1
Difficulty	E	4232.88	0.00	1.00	7368.43	0.00	1	7368.43	0.00	1
Resource	D	961.72	0.06	1.83	3962.99	0.00	1	3972.28	0.00	1
Capacity	M	1967.34	0.00	1.00	4869.80	0.00	1	4869.80	0.00	1
Difficulty	E	2014.86	0.07	1.50	5181.15	0.00	1	5181.15	0.00	1

Table 6.30: The Result of BFP Procedure for maxNPV/minMWT

INS		GA Application			BFP in the Intermediate Stages					
					Without BFP on the Archive			BFP on the Archive		
		ANPV	AMCT	ANNS	ANPV	AMCT	ANNS	ANPV	AMCT	ANNS
Due	D	1385.57	40.63	3.83	4023.63	29.06	1.00	4023.63	29.06	1.00
Date	M	1343.72	40.12	2.83	3235.12	31.12	1.67	3772.65	29.55	1.33
Difficulty	E	855.05	44.96	4.67	4062.93	28.92	1.83	4062.50	28.92	2.00
Lump Sum	D	-1790.16	54.38	10.50	-1073.38	45.68	8.83	165.84	29.31	2.33
Payment	M	1086.08	40.65	3.17	4175.99	28.83	1.83	4205.62	28.74	1.50
Difficulty	E	4244.65	39.93	2.83	7064.32	30.09	1.50	7068.35	30.08	1.50
Resource	D	1129.36	42.16	3.67	4025.36	29.06	1.67	4025.36	29.06	1.67
Capacity	M	2531.07	34.91	2.33	5044.66	25.16	1.17	5044.66	25.16	1.17
Difficulty	E	2153.41	35.91	4.17	5151.87	24.29	1.67	5151.87	24.29	1.67

Table 6.31: The Result of BFP Procedure for maxNPV/minMCT

INS		Without BFP on the Archive				BFP on the Archive			
		ACMAX	ANPV	ARUD	ANNS	ACMAX	ANPV	ARUD	ANNS
Due	D	129.16	-2641.35	10.14	263.80	34.24	2691.25	4.79	19.20
Date	M	138.93	-2865.66	9.57	293.20	35.03	2385.27	5.32	22.20
Difficulty	E	133.33	-2835.01	9.79	270.60	33.99	2794.10	4.68	18.60
Lump Sum	D	133.59	-3458.20	10.68	430.00	47.50	-519.64	9.37	30.20
Payment	M	134.30	-2889.33	9.78	246.00	34.40	2757.05	5.68	23.00
Difficulty	E	127.99	-1834.45	9.17	159.80	35.08	5865.37	5.34	17.40
Resource	D	142.23	-2867.57	9.71	325.00	34.38	2554.33	4.98	20.00
Capacity	M	136.65	-2562.70	9.36	291.60	44.56	3254.22	7.65	25.40
Difficulty	E	127.59	-2503.89	9.35	214.80	61.11	2390.78	10.85	48.60

Table 6.32: The Result of BFP Procedure for minCMAX/maxNPV/minRUD

INS		Without BFP on the Archive				BFP on the Archive			
		ACMAX	ANPV	AMO	ANNS	ACMAX	ANPV	AMO	ANNS
Due	D	94.48	-450.22	-5113.11	89.20	92.37	-32.78	-4971.15	84.40
Date	M	89.17	-491.66	-5191.59	82.40	85.97	122.41	-5076.05	78.40
Difficulty	E	92.78	-639.85	-5086.27	87.20	91.66	-211.79	-5082.06	79.40
Lump Sum	D	97.56	-1908.76	-5217.07	91.80	94.12	-1564.31	-5248.70	90.80
Payment	M	95.76	-649.61	-5031.59	77.80	93.39	-152.55	-5070.82	73.40
Difficulty	E	94.69	863.25	-4493.67	78.40	88.67	1556.52	-4791.37	84.60
Resource	D	94.30	-574.97	-4689.61	75.80	88.32	-60.14	-4991.76	81.80
Capacity	M	90.81	-88.58	-4847.73	90.40	86.80	280.33	-5001.63	95.20
Difficulty	E	89.58	60.44	-4856.91	87.20	89.62	360.93	-4700.45	79.20

Table 6.33: The Result of BFP Procedure for minCMAX/maxNPV/maxMO

these solutions, memory limit becomes a constraint for the algorithm. Therefore, following tables 6.32 and 6.33 show the results of the instances excluding the instances whose number of activity is 240.

## 6.2 Results

After all of the experiments, some insights are gained about the performance of the algorithm, the performance of BFP and the performance of entropy-based divergence application.



### **6.2.1 The test instances generated in Can and Ulusoy [20]**

When BFP is applied only in the final stage, it is observed that BFP shows good performance. For all of the instance sets A, B and C, the solutions obtained without BFP on the archive are significantly improved by applying BFP on this solution set. However, the number of solutions provided is decreased significantly. This situation can be evaluated that as the quality of a solution increases, this solution starts becoming rare and the probability of finding any better solution decreases. Another interpretation can be made that BFP procedure can not explore the neighborhood of the solutions on which BFP is applied.

In addition to simple observation made to determine whether a difference between average Cmax and Npv values exists, the result of paired t-test can be evaluated for this purpose as well. According to the results in Table 6.6, 6.7 and 6.8, there are significant improvements between them for all sets A, B and C since the corresponding t stat values are in rejection regions. That is, all t stat values are either bigger than the corresponding t critical values or less than the negative of the corresponding t critical values. Paired t-test is executed with 0.05 significance level.

BFP is applied in the intermediate stages of the algorithm. In this case, if the quality of the solutions obtained without applying BFP on the archive is compared to the quality of the solutions obtained with pure application of GA, it reveals that BFP in the intermediate stages seems to be effective. In addition, the number of non-dominated solutions are generally decreased, but not seriously. That aspect can be accepted as one of the advantages of BFP in the intermediate stages. Moreover, when BFP is applied in the intermediate stages, the quality of the solutions obtained without BFP on the archive and the solutions obtained with BFP on the archive can be compared. It is observed slight improvements, if at all. This observation implies that improvements introduced by BFP in the intermediate stages do not leave much room for further improvement. The number of non-dominated solutions between these two groups are not significantly different as well.

CPU times should also be evaluated. Average running times of the algorithm in which BFP is applied only in the final stage are longer than those of the algorithm, where BFP is applied in the intermediate stages. While the difference between CPU times is not significant for set A, it becomes very important if set B and C is considered. It means BFP in the intermediate stages is superior than BFP in the final stage in terms of both solution quality and CPU times. Even for set C, since the solution quality of BFP in the intermediate stages is better than those of BFP in the final stage, a small difference between CPU

times can be disregarded. The reason why CPU times of BFP in the intermediate stages is less than those of BFP in the final stage is that the algorithm stops after a couple of BFP applications.

It is interesting to interpret Table 6.12. In this table, the frequency of BFP applications during the implementation of GA are shown. Additionally, the bounds on generations numbers are presented. It should be noted that when pure GA is run, the algorithm runs until the bound on generation numbers without observing its convergence. We should inform that we will study pure GA with convergence application, which will be similar to archive based convergence check.

When entropy-based divergence application is performed, it seems to be inefficient. As stated before, entropy-based divergence application is operated for only set A. It should be recalled that if entropy-based divergence is applied in the algorithm, BFP is used neither for the final stage and nor for the intermediate stages. The results show that the solutions obtained with entropy-based divergence application is worse than the solutions obtained with pure GA application. In Table 6.3, the columns under the heading Without BFP on the Archive show the results when BFP is never used. Thus, it can be said that entropy-based divergence application performs worse than the pure GA. Furthermore, since entropy-based divergence applies some additional operations, it increases CPU times. Although it seems to be ineffective, we would like to study more on it by modifying the procedure. This might become one of our future studies.

By observing the frequencies of activity and mode entropies, it can be seen that the number of times that entropy-based divergence is applied is reasonable. It means that the threshold values are determined well. In addition, it seems that the frequency values of the second level thresholds are larger than those of the first level thresholds. It becomes in line with our expectations.

## **6.2.2 The test instances generated in this thesis**

The test instances generated in this thesis are solved with five different objective combinations, which are maxNPV/minMFT, maxNPV/minMWT, maxNPV/minMCT, minC-MAX/maxNPV/minRUD and minC-MAX/maxNPV/maxMO. In addition, the effect of BFP is observed on these objectives.

### 6.2.2.1 Bi-objective

- maxNPV/minMFT (Maximization of Npv and minimization of mean flow time of the projects): When maxNPV/minMFT is the objective combination, the differences in due date difficulties and resource capacity difficulties of the instances do not effect the solution quality. That is, no pattern in solution quality (the objective values) is observed. It can be interpreted that the differences between difficulty levels might not be significant to be able to create differences on solution qualities. However, the variation in lump sum payment difficulties generally affects the solution quality. It is naturally true that if lump sum payments of the projects are increased, average Npv values of the instances are increased. The other point is that average mean flow time of the projects are also generally affected and decreased. That might be a signal that as lump sum payments increase, the algorithm tries to complete the projects as soon as possible to take advantage of the lump sum payments. As for CPU times, the variation in lump sum payment difficulties are observed to affect the running time. As lump sum payments of the projects are increased, the running times of the algorithm are decreased. This might imply that when lump sum payments increase, the objectives does not become conflicting any more and the problem turns into single objective problem. Any clear pattern like this is not observed in due date difficulties and resource capacity difficulties. It is not observed that average number of non-dominated solutions are significantly different through different difficulties of due dates, resource capacities and lump sum payments.

- maxNPV/minMWT (Maximization of Npv and minimization of mean weighted tardiness of the projects): If maxNPV/minMWT is the objective combination, the difficulty levels of due date and resource capacity do not impact the solution quality. In other words, as due date and resource capacity difficulties are decreased, any pattern in Npv or mean weighted tardiness values is not observed. This can be evaluated with the same reason above. However, the changes in lump sum payments clearly impact the solution quality. As lump sum payments of the projects are increased, average Npv values are increased and mean weighted tardiness values are decreased except for the first instance. Whereas it is obvious that the increase of lump sum payments positively affects the Npv values, it is interesting to observe that mean weighted tardiness values are decreased. This case may be evaluated with the same consideration above. As for the average number of non-dominated solutions, they are generally decreased with the decrease of mean weighted tardiness. It is clear that if mean weighted tardiness becomes zero, then the problem turns into single objective problem. Thus, the average number of non-dominated solutions are

generally one. If CPU times are considered, any pattern is not observed.

- **maxNPV/minMCT** (Maximization of Npv and minimization of mean completion time of the projects): If maxNPV/minMCT is considered as the objective combination, the mean completion times of the projects show an interesting pattern that average completion time values of the medium due date difficulty instances are minimum as compared to other difficulty labels, except for fifth instance. On the other hand, average Npv values are not affected by the variation in due date difficulties. As for the number of solutions provided, medium difficulty instances can find minimum number of solutions when compared to other difficulty labels. Furthermore, as lump sum payments are increased, average Npv values are also clearly increased and average completion times are decreased, except for the first and the second instance. In contrast to all of these findings, the variation in resource capacity difficulties do not influence the solution quality. Moreover, CPU times of instances, which are solved with lump sum payment difficulties, show that as lump sum payments are increased, running times of the algorithm are generally decreased. This case may be evaluated with same reasoning above. As for the average number of non-dominated solutions, any pattern is not observed.

### **6.2.2.2 Triple-objective**

- **minCMAX/maxNPV/minRUD** (Minimization of Cmax, maximization of Npv and minimization of resource usage deviation): In the case that minCMAX/maxNPV/minRUD is considered as the objective combination, the levels of due date difficulty and resource capacity difficulty do not impact the solution quality. That is, no pattern is observed in average Cmax, Npv and the number of non-dominated solutions. This case may be explained with the same reason that the difference between the due date difficulties and resource capacity difficulties are not significant. The increase of lump sum payments causes Cmax values to increase and average deviation values to decrease, especially for the last four instances. Npv values are naturally increased with the decrease of lump sum payment difficulties. It is also interesting to observe that number of solutions are decreased as lump sum payment difficulties are decreased. This case might be explained that as lump sum payments increase, the conflicting relationship between Cmax and Npv is not valid anymore. Thus, domination principle of the algorithm labels less individuals as non-dominated. As for running times, no pattern is observed in CPU times of the algorithm.

- **minCMAX/maxNPV/maxMO** (Minimization of Cmax, maximization of Npv and maximization of minimum outflow): When GA is performed with the objective combi-

nation minCMAX/maxNPV/maxMO, it is observed that the variation in due date does not influence the solution quality, but it impacts the average number of solutions. As due date difficulty increases, the number of solutions increases. Like due date difficulty levels, resource capacity difficulty levels do not impact the solution quality. In addition, lump sum payment difficulty levels affect only Npv values. It should be emphasized in this point that average minimum outflow values are not affected by the variation in lump sum payment difficulties. This case is obvious that maximum outflow of the projects are not affected by the lump sum payments if the way of calculating maximum outflow is considered. Additionally, no pattern is observed in CPU times of the algorithm.

### **6.2.2.3 The evaluation of BFP for current objectives**

As stated before in Section 6.1.2, BFP is applied in the intermediate stages when solving the test instances with the objective combinations maxNPV/minMFT, maxNPV/minMWT and maxNPV/minNCT. It is performed only in the final stage when GA solves them with the objective combinations minCMAX/maxNPV/minRUD and minCMAX/maxNPV/maxMO. The reason for this is that the archive can find better solutions in a couple of generations. It means that the ability of the archive to find better solutions is excellent. One possible reason for this situation is that non-dominated sorting procedure considers three objectives simultaneously and this increases the probability of finding better solutions. Thus, since BFP is applied for improving the solution quality when the algorithm can not find better solutions through predetermined number of generations, it is not preferred for the objective combinations minCMAX/maxNPV/minRUD and minCMAX/maxNPV/maxMO.

It is observed that BFP in the intermediate stages becomes efficient for the objective combinations maxNPV/minMFT, maxNPV/minMWT and maxNPV/minNCT. Significant improvements are observed when the quality of the solutions obtained after the last generation of GA (the solutions under the heading Without BFP on the Archive) and the quality of the solutions obtained with pure GA (the solutions under the heading GA Application) are compared to each other. Furthermore, there are small improvements between the solution quality obtained with BFP in the intermediate stages and BFP on the archive. This observation implies that improvements introduced by BFP in the intermediate stages do not leave much room for further improvement. While these results are enough to claim that BFP in the intermediate stages are efficient for these objectives, CPU times also support the efficiency of the algorithm. The running time of the algorithm in which BFP in the intermediate stages are applied are less than those of the algorithm, which is called pure GA.

It can be said that BFP in the final stage is also efficient for improving the solution quality when minCMAX/maxNPV/minRUD is the objective combination. However, the same claim can not be made as strongly for objective combination minCMAX/maxNPV/maxMO because only a small improvement is observed for this case.

## Chapter 7

### Conclusion

In this thesis, multi-objective multi-project resource constrained project scheduling problem is considered. For this problem, as can be seen in the literature, several exact, heuristic and metaheuristic solution procedures have been developed. Among metaheuristic procedures, GA, SA and TS are observed to be the best algorithms by different studies. Since multi-objective solution requires a different framework in these algorithms, various adjustments have been performed on them. For GA procedure, one of the best multi-objective solution procedure is accepted as NSGA-II proposed by Deb [39]. Therefore, in addition to utilizing NSGA-II in the proposed form, some extensions are proposed and implemented in this thesis.

During implementing the algorithm, the effort is not bounded tightly. In other words, while implementing a part of the algorithm, various forms of the corresponding part are taken into account and tried to be applied, if possible. The reason of this variety is to be able to gain and maintain the efficiency and the flexibility in the algorithm. For this purpose, several objective functions (and objective combinations) are adopted from the literature and some of them are developed in this thesis. While implementing the objective combinations, conflicting objectives must exist in the objective combinations. That is, while an objective gets better, the other objective should get worse. Therefore, a time-based objective and a monetary-based objective are generally considered simultaneously in an objective combination.

There are a couple of instance libraries created by different studies. In this thesis, the instance library called PSPLIB (project scheduling problem library) created by Kolisch and Sprecher [89] is preferred. Since the instances in this library have only time-based information of project and are created for only single project instances, the monetary-based

information of projects are assigned. There are two different cost assignment techniques in this thesis, one of which is implemented for fine-tuning of the parameters, which is conducted with single projects and the other one is applied for creating multi-project instance files from single project instance files. The second cost assignment technique is adopted from Can and Ulusoy [20]. While creating the multi-project instance files, a method is proposed in this thesis. According to this method, three different instance sets with respect to different due date, lump sum payment and resource capacity difficulties are created. In addition, a multi-project instance set created by Can and Ulusoy [20] is utilized as well.

In order to maintain the flexibility of the algorithm, three different crossover mechanisms (one-point crossover, two-point crossover and multi-component uniform order-based crossover) and two different parent selection mechanisms (roulette wheel selection and binary tournament selection) are implemented. Moreover, the way of forming the initial population varies. Two different techniques, which are random initial population and feasible initial population are implemented. Whereas random initial population is implemented as proposed in the literature, the feasible initial population is proposed in this thesis to secure that all individuals are feasible with respect to non-renewable resources.

Fine-tuning of the parameters of GA is executed twice. The first experiment is performed in depth with single project instance files taken from Kolisch and Sprecher [89]. In this experiment, parameter combinations (population size, generation number, crossover rate and mutation rate) and operant combinations (one-point crossover and roulette wheel selection, one-point crossover and binary tournament selection, two-point crossover and roulette wheel selection, two-point crossover and binary tournament selection, multi-component uniform order-based crossover and roulette wheel selection, multi-component uniform order-based crossover and binary tournament selection) are experimented. In order to evaluate the solution quality, hypervolume, maximum spread and the number of non-dominated solutions obtained at the end of implementation of the algorithm are accepted as performance measures. As for the second experiment, it is conducted with multi-project instances. While the main procedure is the same with the previous experiment, some differences exist such as the exclusion of crossover and parent selection mechanisms and the exclusion of crossover and mutation rates. The reason of conducting a new experiment is that population size and generation numbers should be functions of the number of total activities in multi-project network. For both experiments, response surface optimization, which is a statistical method, is utilized for determining the best parameter combinations.



Various mechanisms for maintaining the diversity and improving the solution quality are proposed in this thesis. However, entropy-based divergence application and backward and forward pass procedures are studied in detail. The former one basically depends on the structural similarity of the chromosomes for determining the convergence of the algorithm. In this is the case, then some precautions are taken for both activity positions similarity and assigned mode similarities. The second method is developed so as to improve the solution quality. It improves the solution quality by shifting the activities to the left by using the slacks (forward pass) and shifting the activities to the right (backward pass). It is developed for all objective combinations considered in this thesis.

In computational study, the multi-project instance sets created by Can and Ulusoy [20] are solved with the objective combination minCMAX/maxNPV. BFP procedure is applied both in the final stage and in the intermediate stages to evaluate the affect of this procedure on solution quality. When it is applied in the final stages, the solution quality improves. However, the number of non-dominated solutions decreases because as BFP finds better solutions, much better solutions become rare. An alternative explanation can be that BFP can not explore the broader area. In case that BFP is applied in the intermediate stages, it shows good performance if it is compared to pure GA. In addition, the number of non-dominated solutions do not decrease seriously. If BFP in the intermediate stages is compared to BFP on the archive, then a slight improvement is observed because the good performance of BFP in the intermediate stages does not leave a room for improvement. As for CPU times, the running times of BFP in the intermediate stages are less than those of pure GA and those of BFP in the final stage. This occurs because the algorithm can stop running if any improvement chance is not observed during application of BFP in the intermediate stages. In this point, we should recall that pure GA will be studied with archive-based convergence check. In other words, the algorithm will be able to stop running if it is observed that the archive can not renew itself.

In addition to BFP procedure, entropy-based divergence application is applied as well for a small part of the instance set created by Can and Ulusoy [20]. It is observed that the divergence application is not efficient for improving the solution quality. We will try to improve its mechanism and observe whether it is efficient.

The multi-project instance set created in this thesis are solved with the remaining objective combinations. Since they have different difficulty levels of due date, lump sum payment and resource capacity, the effects of change in difficulties are observed on the solution quality. In addition, BFP procedure is applied in the final stage for some objective combinations and is applied in the intermediate stages for the remaining objective

combinations.

According to the results, the variation in due date and resource capacity difficulties do not affect the solution quality for almost all objective combinations. The reason of this observation may be that difference in the difficulties is not significant to be able to create differences in solution quality. For the objective combinations maxNPV/minMFT, maxNPV/minMWT and maxNPV/minMCT, the differences in lump sum payment impact the solution quality. In other words, as lump sum payments increase, average MFT, MWT and MCT values decrease and average NPV values increase. The reason behind it is that the algorithm tries to complete the projects as soon as possible to take advantage of lump sum payments. If the objective combinations maxNPV/minMFT and maxNPV/minMCT are considered, as lump sum payments increase, the running time of the algorithm decreases probably because of turning of the problem into single objective case. In both the objective combinations maxNPV/minMWT and minCMAX/maxNPV/minRUD, the number of non-dominated solutions decreases when average MWT decreases and lump sum payments increase, respectively. The reasons for both case may be the same that the multi-objective problem turns into single objective problem under the conditions mentioned. In other words, the domination principle of the algorithm can label less individuals as non-dominated. For the last objective combination minCMAX/maxNPV/maxMO, lump sum payment do not affect the average MO because of the nature of MO calculation. As one of future studies, multi-factor analysis of the results will be studied more elaborately.

BFP is applied in the final stage with triple-objective because the archive can renew itself excellently. The reason of this is that triple-objective case facilitates the algorithm to find better solutions easier. Thus, it is not necessary to apply BFP in the intermediate stages when triple-objective considered. For the remaining objective combinations, BFP is applied in the intermediate stages. According to the results, both BFP in the final stage and intermediate stages show good performance.

## Bibliography

- [1] Abbasi, B., Shadrokh, S., and Arkat, J. (2006). Bi-objective resource-constrained project scheduling with robustness and makespan criteria. *Applied Mathematics and Computation*, 180:146.
- [2] Adhau, S., Mittal, M. L., and Mittal, A. (2012). A multi-agent system for distributed multi-project scheduling: An auction-based negotiation approach. *Engineering Applications of Artificial Intelligence*, 25:1738–1751.
- [3] Al-Fawzana, M. and Haouari, M. (2005). A bi-objective model for robust resource-constrained project scheduling. *International Journal of Production Economics*, 96:175–187.
- [4] Alcaraz, J. and Maroto, C. (2001). A robust genetic algorithm for resource allocation in project scheduling. *Annals of Operations Research*, 102:83–109.
- [5] Alcaraz, J., Maroto, C., and Ruiz, R. (2004). Improving the performance of genetic algorithms for the rcps problem. In *Proceedings of the Ninth International Workshop on Project Management and Scheduling*, pages 40–43.
- [6] Arazo, J. A., Pajares, J., and Lopez-Paredes, A. (2010). Simulating the dynamic scheduling of project portfolios. *Simulation Modelling Practice and Theory*, 18:1428–1441.
- [7] Azaron, A., Katagiri, H., and Sakawa, M. (2007). Time-cost trade-off via optimal control theory in markov pert networks. *Annals of Operations Research*, 150:47–64.
- [8] Ballestín, F. and Blanco, R. (2011). Theoretical and practical fundamentals for multi-objective optimisation in resource-constrained project scheduling problems. *Computers & Operations Research*, 38:51–62.

- [9] Bartusch, M., Mhring, R. H., and Radermacher, F. C. (1988). Scheduling project networks with resource constraints and time windows. *Annals of Operations Research*, 16:201–240.
- [10] Blazewicz, J., Lenstra, J. K., and Kan, A. H. G. (1983). Scheduling subject to resource constraints: classification and complexity. *Annals of Operations Research*, 5:11–24.
- [11] Boctor, D. (2008). Project management organization. *Management*, 3:3–9.
- [12] Boctor, F. F. (1990). Some efficient multi-heuristic procedures for resource-constrained project scheduling. *European Journal of Operational Research Volume*, 49:3–13.
- [13] Bomsdorf, F. and Derigs, U. (2008). A model, heuristic procedure and decision support system for solving the movie shoot scheduling problem. *OR Spectrum*, 30:751–772.
- [14] Bouffard, V. and Ferland, J. A. (2007). Improving simulated annealing with variable neighborhood search to solve the resource-constrained scheduling problem. *Journal of Scheduling*, 10:375–386.
- [15] Bouleimen, K. and Lecocq, H. (2003). A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple modes version. *European Journal of Operational Research*, 149:268–281.
- [16] Brinkmann, K. and Neumann, K. (1996). Heuristic procedures for resourceconstrained project scheduling with minimal and maximal time lags: the resourcelevelling and minimum projectduration problems. *Journal of Decision Systems*, 5:129–155.
- [17] Browning, T. R. and Yassine, A. A. (2010). Resource-constrained multi-project scheduling: priority rule performance revisited. *International Journal of Production Economics*, 126:212–228.
- [18] Brucker, P. and Knust, S. (2000). A linear programming and constraint propagation-based lower bound for the rcpsp.
- [19] Brucker, P., Knust, S., Schoo, A., and Thiele, O. (1998). A branch and bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 107:272–288.

- [20] Can, A. and Ulusoy, G. (2010). Multi-project scheduling with 2-stage decomposition.
- [21] Carazo, A. F., Gmez, T., Molina, J., Hernandez-Daz, A. G., Guerrero, F. M., and Caballero, R. (2010). Solving a comprehensive model for multiobjective project portfolio selection. *Computers & Operations Research*, 37:630–639.
- [22] Cesta, A., Oddi, A. H., and Smith, S. F. (2002). A constrained-based method for project scheduling with time windows. *Journal of Heuristics*, 8:109–136.
- [23] Chen, J. and Askin, R. G. (2009). Project selection, scheduling and resource allocation with time dependent returns. *European Journal of Operational Research*, 193:23–34.
- [24] Chen, P. H. and Shahandashti, S. M. (2009). Hybrid of genetic algorithm and simulated annealing for multiple project scheduling with multiple resource constraints. *Automation in Construction*, 18:434–443.
- [25] Chen, V. Y. (1994). A 0-1 goal programming model for scheduling multiple maintenance projects at a copper mine. *European Journal of Operational Research*, 76:176–191.
- [26] Cheng, H., Chiang, T., and Fu, L. (2011). A two-stage hybrid memetic algorithm for multiobjective job shop scheduling. *Expert Systems with Applications*, 38:10983–10998.
- [27] Chiu, H. N. and Tsai, D. M. (2002). An efficient search procedure for the resource-constrained multi-project scheduling problem with discounted cash flows. *Construction Management & Economics*, 20:55–66.
- [28] Cho, J. H. and Kim, Y. D. (1997). A simulated annealing algorithm for resource constrained project scheduling problems. *Journal of the Operational Research Society*, 48:735–744.
- [29] Cochran, J., Horng, S., and Fowler, J. (2003). A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. *Computers & Operations Research*, 30:1087–1102.
- [30] Coffin, M. A. and Taylor III, B. W. (1996). Multiple criteria r&d project selection and scheduling using fuzzy logic. *Computers & Operations Research*, 23:207–220.

- [31] Confessore, G., Giordani, S., and Rismondo, S. (2007). A market-based multi-agent system model for decentralized multi-project scheduling. *Annals of Operations Research*, 150:115–135.
- [32] Cooper, D. (1976). Heuristics for scheduling resource-constrained projects: An experimental investigation. *Management Science*, 22:1186–1194.
- [33] Czyzak, P. and Jaszkiwicz, A. (1998). Pareto simulated annealing a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Analysis*, 7:34–47.
- [34] Davis, E. and Patterson, J. (1975). A comparison of heuristic and optimum solutions in resource-constrained project scheduling. *Management Science*, 21:944–955.
- [35] Davis, K., Stam, A., and Grzybowski, R. (1992). Resource constrained project scheduling with multiple objectives: A decision support approach. *Computers & Operations Research*, 19:657–669.
- [36] De Reyck, B. and Herroelen, W. S. (1998a). A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations. *European Journal of Operational Research*, 111:152–174.
- [37] De Reyck, B. and Herroelen, W. S. (1998b). An optimal procedure for the resource-constrained project scheduling problem with discounted cash flows and generalized precedence relations. *Computers & Operations Research*, 25:1–17.
- [38] De Reyck, B. and Herroelen, W. S. (1999). The multi-mode resource-constrained project scheduling problem with generalized precedence relations. *European Journal of Operational Research*, 119:538–556.
- [39] Deb, K. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197.
- [40] Debels, D., Reyck, B. D., Leus, R., and Vanhoucke, M. (2006). A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. *European Journal of Operational Research*, 169:638–653.
- [41] Demeulemeester, E., Vanhoucke, M., and Herroelen, W. (2003). A random network generator for activity-on-the-node networks. *Journal of Scheduling*, 6:13–34.

- [42] Demeulemeester, E. L. and Herroelen, W. S. (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, 38:1803–1818.
- [43] Demeulemeester, E. L. and Herroelen, W. S. (1997). New benchmark results for the resource-constrained project scheduling problem. *Management Science*, 43:1485–1492.
- [44] Doersch, R. H. and Patterson, J. H. (1977). Scheduling a project to maximize its present value: A zero-one programming approach. *Management Science*, 23:882–889.
- [45] Dorndorf, U., Pesch, E., and Phan-Huy, T. (2000). Time-oriented branch and bound algorithm for resource constrained project scheduling with generalized precedence constraints. *Management Science*, 46:1365–1384.
- [46] Drexler, A. (1991). Scheduling of project networks by job assignments. *Management Science*, 37:1590.
- [47] Drexler, A. and Grnewald, J. (1993). Nonpreemptive multi-mode resource constrained project scheduling. *IIE Transactions*, 25:74–81.
- [48] Elazouni, A. and Abido, M. (2011). Multiobjective evolutionary finance-based scheduling: Individual projects within a portfolio. *Automation in Construction*, 20:755–766.
- [49] Elloumi, S. and Fortemps, P. (2010). A hybrid rank-based evolutionary algorithm applied to multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 205:31–41.
- [50] Esquivel, S., Ferrero, S., Gallard, R., Salto, C., Alfonso, H., and Schtz, M. (2002). Enhanced evolutionary algorithms for single and multiobjective optimization in the job shop scheduling problem. *Knowledge-Based Systems*, 15:13–25.
- [51] Fatemi Ghomi, S. M. T. and Ashjari, B. (2002). A simulation model for multi-project resource allocation. *International Journal of Project Management*, 20:127–130.
- [52] Fest, A., Mhring, R. H., Stork, F., and Uetz, M. (1999). Resource-constrained project scheduling with time windows: A branching scheme based on dynamic release dates. Technical report, Technical University of Berlin.

- [53] Fleischer, M. (2002). The Measure of Pareto Optima: Applications to Multiobjective Metaheuristics. Technical report, Army Research Laboratory.
- [54] Fleszar, K. and Hindi, K. S. (2004). Solving the resource-constrained project scheduling problem by a variable neighbourhood search. *European Journal of Operational Research*, 155:402–413.
- [55] Fowler, J. W., Kim, B., Carlyle, W. M., Gel, E., and Horng, S. (2005). Evaluating solution sets of a posteriori solution techniques for bi-criteria combinatorial optimization problems. *Journal of Scheduling*, 8:75–96.
- [56] Ghoddousi, P., Eshtehardian, E., Jooybanpour, S., and Javanmardi, A. (2013). Multi-mode resource-constrained discrete timecost-resource optimization in project scheduling using non-dominated sorting genetic algorithm. *Automation in Construction*, 30:216–227.
- [57] Goncalves, J. F., Mendes, J. J., and Resende, M. G. (2008). A genetic algorithm for the resource constrained multi-project scheduling problem. *European Journal of Operational Research*, 189:1171–1190.
- [58] Gutjahr, W. J., Katzensteiner, S., Reiter, P., Stummer, C., and Denk, M. (2010). Multi-objective decision analysis for competence-oriented project portfolio selection. *European Journal of Operational Research*, 205:670–679.
- [59] Hanne, T. (2001). *Intelligent Strategies for Meta Multiple Criteria Decision Making*. Kluwer, Boston.
- [60] Hanne, T. and Nickel, S. (2005). A multiobjective evolutionary algorithm for scheduling and inspection planning in software development projects. *European Journal of Operational Research*, 167:663–678.
- [61] Hao, Q., Shen, W., Xue, Y., and Wang, S. (2010). Task network-based project dynamic scheduling and schedule coordination. *Advanced Engineering Informatics*, 24:417–427.
- [62] Hapke, M., J. A. and R., S. (1998). Interactive analysis of multiple-criteria project scheduling problems. *European Journal of Operational Research*, 107:315–324.



- [63] Hapke, M., Jaskiewicz, A., and Slowinski, R. (1997). Fuzzy project scheduling with multiple criteria. fuzzy systems. In *Proceedings of the sixth IEEE international conference*, pages 1277–1282.
- [64] Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics*, 45:733–750.
- [65] Hartmann, S. (2001). Project scheduling with multiple modes: A genetic algorithm. *Annals of Operations Research*, 102:111–135.
- [66] Hartmann, S. (2002). A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics*, 49:433–448.
- [67] Hartmann, S. and Kolisch, R. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 127:394–407.
- [68] Heimerl, C. and Kolisch, R. (2010). Scheduling and staffing multiple projects with a multi-skilled workforce. *OR spectrum*, 32:343–368.
- [69] Herroelen, W. S., Dommelen, P. V., and L., D. E. (1997). Project network models with discounted cash flows a guided tour through recent developments. *European Journal of Operational Research*, 100:97–121.
- [70] Homberger, J. (2007). A multi-agent system for the decentralized resource-constrained multi-project scheduling problem. *International Transactions in Operational Research*, 14:565–589.
- [71] Hyun, C. J., Kim, Y., and Kin, Y. K. (1998). A genetic algorithm for multiple objective sequencing problems in mixed model assembly. *Computers & Operations Research*, 25:675–690.
- [72] Icmeli, O. and Erenguc, S. S. (1994). A tabu search procedure for the resource constrained project scheduling problem with discounted cash flows. *Computers & Operations Research*, 21:841–853.
- [73] Icmeli, O. and Erenguc, S. S. (1996). A branch and bound procedure for the resource constrained project scheduling problem with discounted cash flows. *Management Science*, 42:1395.

- [74] Jaskowski, P. and Sobotka, A. (2006). Multicriteria construction project scheduling method using evolutionary algorithm. *Operational Research*, 6:283–297.
- [75] Jaskiewicz, A. (2002). Genetic local search for multi-objective combinatorial optimization. *European Journal of Operational Research*, 137:50–71.
- [76] Kao, H. P., Wang, B., Dong, J., and Ku, K. C. (2006). An event-driven approach with makespan/cost tradeoff analysis for project portfolio scheduling. *Computers in Industry*, 57:379–397.
- [77] Kazaz, B. and Sepil, C. (1996). Project scheduling with discounted cash flows and progress payments. *The Journal of the Operational Research Society*, 47:1262–1272.
- [78] Khattab, M. M. and Choobineh, F. (1991). A new approach for project scheduling with a limited resource. *International Journal of Production Research*, 29:185–198.
- [79] Kılıç, M., Ulusoy, G., and Şerifoğlu, F. (2008). A bi-objective genetic algorithm approach to risk mitigation in project scheduling. *International Journal of Production Economics*, 112:202–216.
- [80] Kim, K., Yun, Y., Yoon, J., Gen, M., and Yamazaki, G. (2005). Hybrid genetic algorithm with adaptive abilities for resource-constrained multiple project scheduling. *Computers in Industry*, 56:143–160.
- [81] Kim, S. O. and Schniederjans, M. J. (1989). Heuristic framework for the resource constrained multi-project scheduling problem. *Computers & Operations Research*, 16:541–556.
- [82] Knowles, J. and Corne, D. (2002). On metrics for comparing nondominated sets. In *Proceedings of the Congress on Evolutionary Computation*.
- [83] Kochetov, Y. and Stolyar, A. (2003). Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem. In *Proceedings of the third international workshop of computer science and information technologies*, pages 20–32.
- [84] Kogan, K. and Shtub, A. (1999). Scheduling projects with variable-intensity activities: The case of dynamic earliness and tardiness costs. *European Journal of Operational Research*, 118:65–80.

- [85] Kolisch, R. (1996a). Efficient priority rules for the resource constrained project scheduling problem. *Journal of Operations Management*, 14:179–192.
- [86] Kolisch, R. (1996b). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational*, 90:320–333.
- [87] Kolisch, R. (2000). Integrated scheduling, assembly area-and part-assignment for large-scale, make-to-order assemblies. *International Journal of Production Economics*, 64:127–141.
- [88] Kolisch, R. and Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 174:23–37.
- [89] Kolisch, R. and Sprecher, A. (1996). Psplib – a project scheduling problem library. *European Journal of Operational*, 96:205–216.
- [90] Kramer, B. A. and Hwang, C. L. (1991). Resource constrained project scheduling: Modelling with multiple alternatives. *Mathematical and Computer Modelling*, 15:49–63.
- [91] Krger, D. and Scholl, A. (2010). Managing and modelling general resource transfers in (multi-) project scheduling. *OR Spectrum*, 32:369–394.
- [92] Krüger, D. and Scholl, A. (2009). A heuristic solution framework for the resource constrained (multi-) project scheduling problem with sequence-dependent transfer times. *European Journal of Operational Research*, 197:492–508.
- [93] Kumanan, S., Jose, G. J., and Raja, K. (2006). Multi-project scheduling using an heuristic and a genetic algorithm. *The International Journal of Advanced Manufacturing Technology*, 31:360–366.
- [94] Kurtulus, I. and Davis, E. W. (1982). Multi-project scheduling: Categorization of heuristic rules performance. *Management Science*, 28:161–172.
- [95] Lawrence, S. R. and Morton, T. E. (1993). Resource-constrained multi-project scheduling with tardy costs: Comparing myopic, bottleneck, and resource pricing heuristics. *European Journal of Operational Research*, 64:168–187.

- [96] Lee, J. and Kim, Y. (1996). Search heuristics for resource constrained project scheduling. *The Journal of the Operational Research Society*, 47:678–689.
- [97] Leon, V. and Balakrishnan, R. (1995). Strength and adaptability of problem-space based neighborhoods for resource-constrained scheduling. *OR Spektrum*, 17:173–182.
- [98] Li, K. Y. and Willis, J. (1992). An iterative scheduling technique for resource-constrained project scheduling. *European Journal of Operational Research*, 56:370–379.
- [99] Lova, A., Maroto, C., and Tormos, P. (2000). A multicriteria heuristic method to improve resource allocation in multiproject scheduling. *European Journal of Operational Research*, 127:408–424.
- [100] Mansouri, S. A. (2005). A multi-objective genetic algorithm for mixed-model sequencing on jit assembly lines. *European Journal of Operational Research*, 167:696–716.
- [101] Mingozzi, A., Maniezzo, V., Ricciardelli, S., and Bianco, L. (1998). An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management Science*, 44:714–729.
- [102] Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24:1097–1100.
- [103] Mohanty, R. and Siddiq, M. (1989). Multiple projects - multiple resources constrained scheduling: A multiobjective analysis. *Engineering Costs and Production Economics*, 18:83–92.
- [104] Mori, M. and Tseng, C. C. (1997). A genetic algorithm for multi-mode resource constrained project scheduling problem. *European Journal of Operational Research*, 100:134–141.
- [105] Myers, R. and Montgomery, D. (1995). *Response surface methodology: Process and product optimization using designed experiments*. John Wiley, New York.
- [106] Nabrzyski, J. and Węglarz, J. (1995). On an expert system with tabu search for multiobjective project scheduling. In *Proceedings of INRIA/IEEE symposium on emerging technologies and factory automation III*, pages 87–94.

- [107] Najafi, A. A. and Niaki, S. T. A. (2006). A genetic algorithm for resource investment problem with discounted cash flows. *Applied Mathematics and Computation*, 183:1057–1070.
- [108] Naphade, K. S., Wu, S. D., and Storer, R. H. (1997). Problem space search algorithms for resource-constrained project scheduling. *Annals of Operations Research*, 70:307–326.
- [109] Neumann, K. and Zhan, Z. (1995). Heuristics for the minimum project-duration problem with minimal and maximal time-lags under fixed resource constraints. *Journal of Intelligent Manufacturing*, 6:145–154.
- [110] Neumann, K. and Zimmermann, J. (2000). Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints. *European Journal of Operational Research*, 127:425–443.
- [111] Nikulin, Y. and Drexl, A. (2010). Theoretical aspects of multicriteria flight gate scheduling: deterministic and fuzzy models. *Journal of Scheduling*, 13:261–280.
- [112] Nsakanda, A. L., Price, W. L., Diaby, M., and Gravel, M. (2007). Ensuring population diversity in genetic algorithms: A technical note with application to the cell formation problem. *European Journal of Operational Research*, 178:634–638.
- [113] Nudtasomboon, N. and Randhawa, S. U. (1997). Resource-constrained project scheduling with renewable and non-renewable resources and time-resource tradeoffs. *Computers and Industrial Engineering*, 32:227–242.
- [114] Osman, M. S., Abo-Sinna, M. A., and Mousa, A. A. (2005). An effective genetic algorithm approach to multiobjective resource allocation problems (moraps). *Applied Mathematics and Computation*, 163:755–768.
- [115] Ozdamar, L. and Ulusoy, G. (1996). A note on an iterative forward/backward scheduling technique with reference to a procedure by li and willis. *European Journal of Operational Research*, 89:400–407.
- [116] Ozmehmet Tasan, S. and Gen, M. (2013). An integrated selection and scheduling for disjunctive network problems. *Computers & Industrial Engineering*, 65:65–76.

- [117] Pan, H. and Yeh, C.-H. (2003). *Knowledge-Based Intelligent Information and Engineering Systems*, volume 2773, chapter A metaheuristic approach to fuzzy project scheduling, pages 1081–1087. Springer, Amsterdam.
- [118] Pritsker, A. A. B., Waiters, L. J., and Wolfe, P. M. (1969). Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science*, 16:93–108.
- [119] Ramirez Palencia, A. E. and Meja Delgadillo, G. E. (2012). A computer application for a bus body assembly line using genetic algorithms. *International Journal of Production Economics*, 140:431–438.
- [120] Rojas, I., González, J., Pomares, H., Merelo, J. J., Castillo, P. A., and Romero, G. (2002). Statistical analysis of the main parameters involved in the design of a genetic algorithm. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 32:31–37.
- [121] Sampson, S. and Weiss, E. (1993). Local search techniques for the generalized resource constrained project scheduling problem. *Naval Research Logistics*, 40:665–675.
- [122] Sayin, S. (2000). Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming. *Mathematical Programming*, 87:543–560.
- [123] Selle, T. and Zimmermann, J. (2003). A bidirectional heuristic for maximizing the net present value of large-scale projects subject to limited resources. *Naval Research Logistics*, 50:130–148.
- [124] Sepil, C. and Ortac, N. (1997). Performance of the heuristic procedures for constrained projects with progress payments. *The Journal of the Operational Research Society*, 48:1123–1130.
- [125] Shadrokh, S. and Kianfar, F. (2007). A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty. *European Journal of Operational Research*, 181:86–101.
- [126] Shtub, A., LeBlanc, L. J., and Cai, Z. (1996). Scheduling programs with repetitive projects: a comparison of a simulated annealing, a genetic and a pair-wise swap algorithm. *European Journal of Operational Research*, 88:124–138.

- [127] Slowinski, R. (1980). Two approaches to problems of resource allocation among project activities – a comparative study. *The Journal of the Operational Research Society*, 31:711–723.
- [128] Slowinski, R., Soniewicki, B., and Węglarz, J. (1994). Dss for multiobjective project scheduling. *European Journal of Operational Research*, 79:220–229.
- [129] Sprecher, A., Hartmann, S., and A., D. (1997). An exact algorithm for project scheduling with multiple modes. *OR Spektrum*, 19:195–203.
- [130] Steuer, R. E. (1986). *Multiple criteria optimization: Theory, computation, and optimization*. Wiley, New York.
- [131] Sun, H. and Ma, T. (2005). A packing-multiple-boxes model for r&d project selection and scheduling. *Technovation*, 25:1355–1361.
- [132] Talbot, F. B. (1982). Resource constrained project scheduling with time-resource trade-offs: The nonpreemptive case. *Management Science*, 28:1197–1210.
- [133] Tamaki, H., Nishino, E., and Abe, S. (1999). A genetic algorithm approach to multi-objective scheduling problems with earliness and tardiness penalties. In *Proceedings of the 1999 Congress on (Vol. 1)*. IEEE., pages 20–32.
- [134] Tiwari, V., Patterson, J. H., and Mabert, V. A. (2009). Scheduling projects with heterogeneous resources to meet time and quality objectives. *European Journal of Operational Research*, 193:780–790.
- [135] Ulusoy, G. and Cebelli, S. (2000). An equitable approach to the payment scheduling problem in project management. *European Journal of Operational Research*, 127:262–278.
- [136] Ulusoy, G. and Ozdamar, L. (1989). Heuristic performance and network/resource characteristics in resource-constrained project scheduling. *Journal of the Operational Research Society*, 40:1145–1152.
- [137] Ulusoy, G. and Ozdamar, L. (1994). A constraint-based perspective in resource constrained project scheduling. *International Journal of Production Research*, 32:693–705.

- [138] Ulusoy, G. and Ozdamar, L. (1995). A heuristic scheduling algorithm for improving the duration and net present value of a project. *International Journal of Operations & Production Management*, 15:89–98.
- [139] Ulusoy, G., Sivrikaya-Serifoglu, F., and Sahin, S. (2001). Four payment models for the multi-mode resource constrained project scheduling problem with discounted cash flows. *Annals of Operations Research*, 102:237–261.
- [140] V., V., Ballestin, F., and Quintanilla, M. S. (2003). A hybrid genetic algorithm for the RCPSP. Technical report, Department of Statistics and Operations Research, University of Valencia.
- [141] Van, V. D. (1999). Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Master’s thesis, School of Engineering of the Air Force Institute of Technology.
- [142] Vanhoucke, M., Coelho, J. S., Debels, D., Maenhout, B., and Tavares, L. V. (2008). An evaluation of the adequacy of project network generators with systematically sampled networks. *European Journal of Operational Research*, 187:511–524.
- [143] Vanhoucke, M., Demeleumeester, E., and Herroelen, W. (2001a). An exact procedure for the resource-constrained weighted earliness-tardiness project scheduling problem. *Annals of Operations Research*, 102:179–196.
- [144] Vanhoucke, M., Demeleumeester, E., and Herroelen, W. (2001b). On maximizing the net present value of a project under renewable resource constraints. *Management Science*, 47:1113–1121.
- [145] Varma, V. A., Pekny, J. F., Blau, G. E., and Reklaitis, G. V. (2008). A framework for addressing stochastic and combinatorial aspects of scheduling and resource allocation in pharmaceutical r&d pipelines. *Computers & Chemical Engineering*, 32:1000–1015.
- [146] Viana, A. and de Sousa, J. P. (2000). Using metaheuristics in multiobjective resource constrained project scheduling. *European Journal of Operational Research*, 120:359–374.
- [147] Voß, S. and Witt, A. (2007). Hybrid flow shop scheduling as a multi-mode multi-project scheduling problem with batching requirements: A real-world application. *International Journal of Production Economics*, 105:445–458.



- [148] Wang, H., Lin, D., and Li, M. (2005). A competitive genetic algorithm for resource constrained project scheduling problem. In *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, pages 2945–2949.
- [149] Xiong, J., Chen, Y. W., Yang, K. W., Zhao, Q. S., and Xing, L. N. (2012). A hybrid multiobjective genetic algorithm for robust resource-constrained project scheduling with stochastic durations. *Mathematical Problems in Engineering*.
- [150] Yamashita, D. S., Armentano, V. A., and Laguna, M. (2006). Scatter search for project scheduling with resource availability cost. *European Journal of Operational Research*, 169:623–637.
- [151] Yannibelli, V. and Amandi, A. (2013). Hybridizing a multi-objective simulated annealing algorithm with a multi-objective evolutionary algorithm to solve a multi-objective project scheduling problem. *Expert Systems with Applications*, 40:2421–2434.
- [152] Zitzler, E. (1999). Evolutionary algorithms for multiobjective optimization: methods and applications. Master’s thesis, Swiss Federal Institute of Technology (ETH).
- [153] Zitzler, E., Laumanns, M., and Thiele, L. (2003a). SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical report, Swiss Federal Institute of Technology (ETH) Zurich.
- [154] Zitzler, E. and Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms a comparative case study. In *Proceedings of the international conference on parallel problem solving from nature*, pages 292–304.
- [155] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., and Grunert da Fonseca, V. (2003b). Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7:117–132.

# Appendices

## Appendix A

# Results of Fine-Tuning Experiments for Single Projects

Abbreviations for the tables:

- **INS:** Instance
- **OB:** One-Point Crossover and Binary Tournament Selection
- **OR:** One-Point Crossover and Roulette Wheel Selection
- **TB:** Two-Point Crossover and Binary Tournament Selection
- **TR:** Two-Point Crossover and Roulette Wheel Selection
- **MB:** Multi Component Uniform Order-Based Crossover and Binary Tournament Selection
- **MR:** Multi Component Uniform Order-Based Crossover and Roulette Wheel Selection
- **Pop:** Population size or coefficient
- **Gen:** Generation number of coefficient
- **Mut:** Mutation rate
- **Crs:** Crossover rate
- **Des:** Desirability
- **Act:** Activities
- **Freq:** Frequency

INS	Pop	Gen	Mut	Crs	Des	Pop	Gen	Mut	Crs	Des
	OB					OR				
10(1)	80	25	0,25	0,8	0,496	60	25	0,25	0,9	0,608
10(2)	100	25	0,25	0,7	0,849	80	25	0,21	0,6	0,848
10(3)	100	25	0,25	0,7	0,786	60	50	0,25	0,9	0,741
10(4)	100	50	0,25	1	0,817	100	100	0,25	0,6	0,858
10(5)	80	50	0,25	0,7	0,789	100	100	0,25	0,7	0,756
20(1)	100	150	0,13	0,6	0,693	100	150	0,17	0,7	0,694
20(2)	100	150	0,21	0,9	0,570	80	125	0,25	0,8	0,54
20(3)	60	75	0,21	0,8	0,644	60	125	0,25	1	0,674
20(4)	60	150	0,21	0,8	0,673	100	150	0,21	0,7	0,663
20(5)	100	75	0,25	0,9	0,648	100	150	0,21	1	0,637
30(1)	100	125	0,13	0,9	0,642	100	150	0,13	0,8	0,598
30(2)	100	150	0,17	1	0,645	100	150	0,13	1	0,674
30(3)	100	100	0,17	1	0,564	100	150	0,25	1	0,557
30(4)	60	125	0,21	0,8	0,610	100	150	0,13	1	0,636
30(5)	100	125	0,13	0,9	0,514	100	100	0,05	1	0,615
	TB					TR				
10(1)	80	50	0,25	0,9	0,567	100	75	0,25	0,6	0,54
10(2)	100	25	0,25	0,6	0,824	100	25	0,25	0,8	0,824
10(3)	100	25	0,25	0,6	0,763	20	25	0,25	0,8	0,745
10(4)	100	50	0,25	1	0,761	100	100	0,25	0,6	0,824
10(5)	80	50	0,21	0,7	0,75	100	75	0,25	0,8	0,791
20(1)	80	150	0,17	0,9	0,714	80	150	0,13	0,8	0,671
20(2)	100	150	0,17	0,7	0,54	60	150	0,21	1	0,61
20(3)	80	75	0,21	0,7	0,69	60	100	0,25	1	0,611
20(4)	60	100	0,25	1	0,656	100	125	0,21	0,9	0,639
20(5)	100	50	0,17	0,9	0,575	80	150	0,25	0,8	0,655
30(1)	60	125	0,17	0,9	0,578	80	150	0,05	0,9	0,616
30(2)	100	125	0,21	0,9	0,584	100	100	0,09	1	0,676
30(3)	80	150	0,25	1	0,53	100	150	0,21	1	0,536
30(4)	80	75	0,13	1	0,609	100	150	0,09	1	0,629
30(5)	100	150	0,09	0,8	0,607	100	150	0,05	0,9	0,545
	MB					MR				
10(1)	80	25	0,17	0,9	0,53	100	25	0,21	0,9	0,58
10(2)	100	25	0,25	0,7	0,849	100	25	0,25	0,9	0,817
10(3)	80	75	0,25	0,6	0,734	100	25	0,21	0,6	0,776
10(4)	100	75	0,25	1	0,772	60	100	0,25	0,8	0,836
10(5)	100	75	0,25	0,8	0,777	80	75	0,25	0,7	0,785
20(1)	100	125	0,21	1	0,636	80	75	0,13	0,7	0,658
20(2)	80	50	0,13	0,8	0,567	100	150	0,25	0,7	0,56
20(3)	100	50	0,17	1	0,619	80	125	0,21	0,7	0,686
20(4)	60	100	0,25	0,8	0,687	100	100	0,25	1	0,619
20(5)	100	50	0,13	0,8	0,7	100	150	0,25	0,9	0,681
30(1)	100	100	0,09	0,6	0,568	80	150	0,13	1	0,624
30(2)	100	150	0,21	0,9	0,563	80	150	0,13	0,8	0,533
30(3)	80	150	0,21	0,9	0,572	80	75	0,17	0,7	0,531
30(4)	100	125	0,17	0,8	0,583	100	125	0,09	0,8	0,572
30(5)	100	125	0,13	0,8	0,581	100	125	0,05	0,7	0,536

Table A.1: Result of the Experiment for Single Projects - 1

Operant Combination	# of Act	Pop	Gen	Mut	Crs	Des
OB	10	100	25	0,25	0,7	0,849
	20	100	150	0,21	0,9	0,57
	30	100	125	0,13	0,9	0,642
OR	10	60	50	0,25	0,9	0,741
	20	100	150	0,21	0,7	0,663
	30	100	150	0,13	1	0,674
TB	10	100	25	0,25	0,6	0,824
	20	80	150	0,17	0,9	0,714
	30	100	125	0,21	0,9	0,584
TR	10	100	75	0,25	0,8	0,791
	20	80	150	0,25	0,8	0,655
	30	100	150	0,09	1	0,629
MB	10	100	75	0,25	0,8	0,777
	20	100	50	0,13	0,8	0,7
	30	100	125	0,17	0,8	0,583
MR	10	100	25	0,25	0,9	0,817
	20	100	150	0,25	0,7	0,56
	30	80	150	0,13	0,8	0,533

Table A.2: Result of the Experiment for Single Projects - 2

Operant Combination	# of Act	Mean Des	Mean Rank	Freq
OB	10	0,77889	3,67724	93
	20	0,75099	3,21818	105
	30	0,71215	3,37273	94
OR	10	0,80084	3,44403	89
	20	0,77619	3,01455	145
	30	0,75293	2,95273	149
TB	10	0,79040	3,62873	89
	20	0,70731	3,71273	68
	30	0,70024	3,46727	92
TR	10	0,81405	3,29478	100
	20	0,75072	3,28364	101
	30	0,71528	3,32000	115
MB	10	0,77055	3,77425	56
	20	0,65810	4,23818	46
	30	0,63866	4,04727	40
MR	10	0,78287	3,54291	99
	20	0,71741	3,59091	85
	30	0,65782	3,89091	58

Table A.3: Result of the Experiment for Single Projects - 3

## Appendix B

# Results of Fine-Tuning Experiments for Multi-Project

Abbreviations in the tables:

- **minC<sub>MAX</sub>/maxNPV**: Minimization of C<sub>max</sub> and Maximization of Npv
- **maxNPV/minMFT**: Maximization of Npv and Minimization of Average Flow Time of the Projects
- **maxNPV/minMWT**: Maximization of Npv and Minimization of Weighted Tardiness of the Projects
- **maxNPV/minMCT**: Maximization of Npv and Minimization of Average Completion Time of the Projects
- **minC<sub>MAX</sub>/maxNPV/minRUD**: Minimization of C<sub>max</sub>, Maximization of Npv and Minimization of Resource Usage Deviation
- **minC<sub>MAX</sub>/maxNPV/maxMO**: Minimization of C<sub>max</sub>, Maximization of Npv and Maximization of Minimum Outflow
- **LS Payment**: Lump sum payment
- **Proj**: Projects
- **Clos**: Closeness
- **CPU**: Processing time (seconds)

Other abbreviations used in this appendix is defined in Appendix A.

# of Proj	# of Act	Sum of Act	minCMAX/maxNPV		maxNPV/minMFT	
			Pop	Gen	Pop	Gen
10	10	100	1.25	2.50	0.75	2.00
10	12	120	1.50	2.00	1.25	3.00
10	14	140	1.50	3.00	1.25	3.00
10	10	100	1.50	2.00	1.00	2.00
10	12	120	1.00	2.50	1.00	1.50
10	14	140	1.50	2.00	1.25	3.00
10	10	100	1.50	3.00	1.25	3.00
10	12	120	1.25	2.00	1.25	3.00
10	14	140	1.25	2.50	1.50	3.00
# of Proj	# of Act	Sum of Act	maxNPV/minMWT		maxNPV/minMCT	
			Pop	Gen	Pop	Gen
10	10	100	1.25	1.00	1.00	3.00
10	12	120	0.75	1.50	1.50	2.50
10	14	140	1.00	3.00	1.50	3.00
10	10	100	0.75	2.00	0.75	3.00
10	12	120	0.75	3.00	1.25	3.00
10	14	140	1.00	2.50	1.25	2.50
10	10	100	1.50	1.00	1.00	1.00
10	12	120	1.25	1.00	1.50	3.00
10	14	140	1.50	2.50	1.25	2.50
# of Proj	# of Act	Sum of Act	minCMAX/maxNPV/minRUD		minCMAX/maxNPV/maxMO	
			Pop	Gen	Pop	Gen
10	10	100	1.50	2.50	1.50	2.50
10	12	120	1.25	2.00	1.50	3.00
10	14	140	1.25	2.50	1.50	2.50
10	10	100	1.50	3.00	1.25	3.00
10	12	120	1.50	3.00	1.50	3.00
10	14	140	1.00	2.50	1.50	3.00
10	10	100	1.50	2.00	1.25	2.00
10	12	120	1.25	3.00	1.50	2.50
10	14	140	1.50	3.00	1.50	2.50

Table B.1: Result of the Experiment for Multiple Projects - 1

# of Proj	# of Act	Sum of Act	minCMAX/maxNPV		maxNPV/minMFT		maxNPV/minMWT		maxNPV/minMCT		minCMAX/maxNPV/minRUD		minCMAX/maxNPV/maxMO							
			Pop	Gen	Des	Pop	Gen	Des	Pop	Gen	Des	Pop	Gen	Des	Pop	Gen	Des			
10	10	100	1.25	2.50	0.48	0.75	2.00	0.47	1.25	1.00	0.36	1.00	3.00	0.44	0.77	1.50	2.50	0.65		
			1.50	2.00	0.39	1.50	2.00	0.44	0.75	1.50	0.35	1.25	1.50	0.32	1.25	3.00	0.74	1.25	3.00	0.64
			1.00	1.50	0.38	1.50	2.50	0.43	0.75	1.50	0.22	1.50	3.00	0.32	1.25	3.00	0.71	1.50	2.00	0.62
10	12	120	1.00	1.50	0.34	1.00	2.50	0.39	1.25	1.50	0.20	0.75	2.50	0.32	1.50	3.00	0.71	1.50	3.00	0.61
			1.50	2.00	0.31	1.00	1.50	0.37	1.00	2.50	0.19	0.75	1.00	0.30	1.25	2.50	0.71	1.00	3.00	0.59
			1.50	3.00	0.40	1.25	3.00	0.46	0.75	1.50	0.31	1.50	2.50	0.45	1.25	2.00	0.74	1.50	3.00	0.77
10	14	140	1.50	2.50	0.36	1.00	3.00	0.43	1.00	1.50	0.21	1.50	3.00	0.42	1.00	3.00	0.74	1.25	3.00	0.75
			1.25	2.00	0.36	1.25	2.00	0.30	1.50	1.00	0.17	1.25	2.50	0.39	1.50	2.50	0.74	0.75	2.50	0.75
			1.00	2.50	0.33	1.25	2.50	0.29	1.25	1.00	0.11	0.75	2.50	0.33	1.25	3.00	0.74	0.75	3.00	0.70
10	10	100	1.25	2.50	0.32	1.25	1.50	0.28	1.25	3.00	0.11	0.75	3.00	0.32	1.50	2.00	0.73	1.50	2.50	0.70
			1.50	3.00	0.27	1.25	3.00	0.21	1.00	3.00	0.18	1.50	3.00	0.29	1.25	2.50	0.81	1.50	2.50	0.81
			1.50	2.50	0.22	1.50	3.00	0.17	1.50	3.00	0.16	1.50	2.00	0.22	1.50	3.00	0.80	1.50	3.00	0.80
10	10	100	1.25	2.00	0.21	1.00	3.00	0.14	1.50	2.50	0.13	1.50	2.50	0.20	1.50	2.00	0.79	1.50	3.00	0.77
			1.25	2.50	0.20	1.00	2.00	0.13	1.50	2.00	0.13	1.25	3.00	0.17	0.75	3.00	0.78	1.25	2.00	0.75
			1.50	2.00	0.16	1.25	2.00	0.13	0.75	3.00	0.11	1.00	3.00	0.16	1.25	3.00	0.77	1.25	2.50	0.75
10	10	100	1.50	2.00	0.45	1.00	2.00	0.34	0.75	2.00	0.35	0.75	3.00	0.33	1.50	3.00	0.66	1.25	3.00	0.72
			1.00	2.50	0.38	0.75	2.50	0.29	1.50	1.00	0.29	1.50	1.50	0.32	1.25	2.50	0.65	1.50	3.00	0.69
			1.00	3.00	0.36	1.25	3.00	0.27	0.75	1.00	0.28	0.75	1.50	0.30	1.50	2.00	0.64	1.50	2.50	0.69
10	12	120	1.00	2.00	0.35	1.50	1.00	0.27	1.00	1.00	0.27	0.75	1.00	0.28	1.25	3.00	0.61	1.50	1.50	0.68
			0.75	2.50	0.34	1.25	2.00	0.27	1.25	1.00	0.16	1.50	3.00	0.25	1.50	1.00	0.61	1.50	1.50	0.68
			1.00	2.50	0.34	1.25	2.00	0.27	1.25	1.00	0.16	1.50	3.00	0.25	1.50	1.00	0.61	1.00	3.00	0.68
10	10	100	1.00	2.50	0.34	1.00	1.50	0.33	0.75	3.00	0.32	1.25	3.00	0.42	1.50	3.00	0.72	1.50	3.00	0.74
			1.25	1.50	0.32	1.50	2.00	0.32	0.75	2.00	0.20	1.50	3.00	0.38	1.00	3.00	0.72	1.25	3.00	0.73
			1.50	3.00	0.29	1.50	1.50	0.31	1.00	1.50	0.16	1.25	2.00	0.37	1.25	3.00	0.72	1.50	2.50	0.72
10	14	140	0.75	1.50	0.28	1.50	3.00	0.30	0.75	1.00	0.16	1.50	2.50	0.37	1.25	2.50	0.70	1.50	2.00	0.70
			1.00	2.00	0.23	1.00	2.50	0.29	0.75	1.50	0.11	0.75	1.00	0.34	1.50	2.50	0.68	1.50	2.00	0.67
			1.50	2.00	0.23	1.00	2.50	0.29	0.75	1.50	0.11	0.75	1.00	0.34	1.50	2.50	0.68	1.50	2.50	0.67
10	10	100	1.50	2.00	0.34	1.25	3.00	0.29	1.00	2.50	0.25	1.25	2.50	0.29	1.00	2.50	0.80	1.50	3.00	0.75
			1.50	3.00	0.33	1.50	2.50	0.24	1.25	1.50	0.17	1.00	3.00	0.25	1.00	3.00	0.80	1.50	2.50	0.72
			1.00	2.50	0.29	1.50	3.00	0.22	1.50	3.00	0.15	1.50	3.00	0.24	0.75	3.00	0.76	1.25	1.50	0.69
10	10	100	0.75	2.50	0.28	1.25	2.50	0.19	1.25	2.50	0.14	1.00	2.50	0.24	1.50	1.50	0.75	1.25	3.00	0.67
			1.25	3.00	0.24	1.50	2.00	0.16	1.00	3.00	0.13	1.50	2.00	0.20	1.25	2.00	0.74	0.75	2.50	0.67
			1.50	3.00	0.24	1.50	2.00	0.16	1.00	3.00	0.13	1.50	2.00	0.20	1.25	2.00	0.74	0.75	2.50	0.67
10	10	100	1.50	3.00	0.36	1.25	3.00	0.37	1.50	1.00	0.31	1.00	1.00	0.32	1.50	2.00	0.73	1.25	2.00	0.77
			1.50	1.00	0.33	1.00	1.00	0.37	0.75	1.00	0.27	0.75	1.50	0.25	1.50	2.50	0.73	1.50	3.00	0.76
			1.25	2.50	0.33	1.00	2.00	0.33	1.00	1.50	0.15	1.00	1.50	0.25	1.50	3.00	0.70	1.50	2.50	0.71
10	12	120	1.50	2.50	0.30	0.75	2.50	0.29	1.00	1.00	0.10	1.00	2.00	0.24	1.00	3.00	0.69	1.25	3.00	0.70
			0.75	2.00	0.28	1.00	1.50	0.28	1.25	1.50	0.10	1.25	2.00	0.23	1.25	1.50	0.68	1.00	2.50	0.67
			1.25	2.00	0.28	1.00	1.50	0.28	1.25	1.50	0.10	1.25	2.00	0.23	1.25	1.50	0.68	1.00	2.50	0.67
10	14	140	1.25	2.00	0.49	1.25	3.00	0.57	1.25	1.00	0.22	1.50	3.00	0.35	1.25	3.00	0.74	1.50	2.50	0.79
			1.50	3.00	0.48	1.50	3.00	0.48	1.50	2.50	0.15	1.00	2.00	0.36	1.50	2.00	0.70	1.25	2.50	0.79
			1.50	2.50	0.44	1.00	3.00	0.44	0.75	2.00	0.14	1.00	2.50	0.33	1.50	2.50	0.67	1.50	3.00	0.77
10	10	100	1.00	2.50	0.38	1.25	2.00	0.44	0.75	1.00	0.13	0.75	2.50	0.30	1.25	2.00	0.67	1.00	3.00	0.75
			1.25	1.50	0.37	0.75	2.50	0.39	0.75	1.50	0.12	1.25	2.00	0.29	1.00	2.00	0.66	1.25	3.00	0.74
			1.25	1.50	0.37	0.75	2.50	0.39	0.75	1.50	0.12	1.25	2.00	0.29	1.00	2.00	0.66	1.25	3.00	0.74
10	10	100	1.25	2.50	0.20	1.50	3.00	0.29	1.50	2.50	0.15	1.25	2.50	0.28	1.50	3.00	0.81	1.50	2.50	0.80
			1.50	2.50	0.20	1.50	2.50	0.23	1.25	2.50	0.14	1.50	2.50	0.28	1.25	2.50	0.81	1.25	3.00	0.75
			1.25	1.50	0.19	1.25	3.00	0.21	1.50	2.00	0.13	1.50	3.00	0.26	1.50	2.00	0.79	1.50	3.00	0.75
10	10	100	1.50	3.00	0.17	1.25	2.50	0.19	0.75	3.00	0.11	0.75	3.00	0.22	0.75	2.00	0.77	1.50	1.00	0.73
			1.00	2.50	0.17	1.00	3.00	0.15	1.00	2.50	0.10	1.00	3.00	0.21	1.50	2.50	0.76	0.75	2.50	0.72
			1.00	2.50	0.17	1.00	3.00	0.15	1.00	2.50	0.10	1.00	3.00	0.21	1.50	2.50	0.76	0.75	2.50	0.72

Table B.2: Result of the Experiment for Multiple Projects - 2



# of Proj	# of Act	Sum of Act	minCMAX/maxNPV			maxNPV/minMFT			maxNPV/minMWT			maxNPV/minMCT			minCMAX/maxNPV/minRUD			minCMAX/maxNPV/maxMO								
			Pop	Gen	Clos	Des	Pop	Gen	Clos	Des	Pop	Gen	Clos	Des	Pop	Gen	Clos	Des	Pop	Gen	Clos	Des				
10	10	100	1.25	2.50	0.11	0.48	1.25	2.00	0.26	0.48	1.25	1.00	0.25	0.36	1.00	3.00	0.16	0.45	1.50	2.50	0.10	0.77	1.50	2.50	0.07	0.66
10	12	120	1.50	2.00	0.11	0.41	1.25	3.00	0.11	0.47	1.50	1.50	0.24	0.31	1.50	2.50	0.17	0.45	1.25	2.00	0.16	0.75	1.50	3.00	0.07	0.77
10	14	140	1.50	3.00	0.15	0.27	1.25	3.00	0.11	0.21	1.00	3.00	0.26	0.18	1.50	3.00	0.16	0.29	1.25	2.50	0.11	0.82	1.50	2.50	0.07	0.82
10	10	100	1.50	2.00	0.11	0.45	1.00	2.00	0.19	0.34	0.75	3.00	0.23	0.35	0.75	3.00	0.22	0.33	1.50	3.00	0.11	0.66	1.25	3.00	0.12	0.73
10	12	120	1.00	2.50	0.18	0.35	1.00	1.50	0.26	0.33	0.75	3.00	0.29	0.32	1.25	3.00	0.13	0.42	1.50	3.00	0.11	0.72	1.50	3.00	0.07	0.74
10	14	140	1.50	2.00	0.11	0.34	1.25	3.00	0.11	0.29	1.00	2.50	0.21	0.25	1.25	2.50	0.14	0.29	1.00	2.50	0.18	0.81	1.50	3.00	0.07	0.75
10	10	100	1.50	3.00	0.15	0.36	1.25	3.00	0.11	0.37	1.50	1.00	0.30	0.31	1.00	1.00	0.36	0.32	1.50	2.00	0.15	0.74	1.25	2.00	0.18	0.77
10	12	120	1.25	2.00	0.12	0.49	1.25	3.00	0.11	0.57	1.25	1.00	0.25	0.22	1.50	3.00	0.16	0.36	1.25	3.00	0.12	0.74	1.50	2.50	0.07	0.79
10	14	140	1.25	2.50	0.11	0.22	1.50	3.00	0.18	0.29	1.50	2.50	0.27	0.15	1.25	2.50	0.14	0.28	1.50	3.00	0.11	0.81	1.50	2.50	0.07	0.80

Table B.3: Result of the Experiment for Multiple Projects - 3