Scheduling of Resource Constrained Projects via Genetic Algorithm with Multiple Skills of Resources in Multi-Project Environment

by

Ayşe Bengi Doğan

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Industrial Engineering

KOÇ
ÜNİVERSİTESİ

October 20, 2023

Scheduling of Resource Constrained Projects via Genetic Algorithm with Invasive Weed Optimization Reproduction Considering the Multiple Skills of Resources in Multi-Project Environment

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Ayşe Bengi Doğan

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Ceyda Oğuz (Advisor)

_____

Prof. Ümit Bilge

_____

Prof. Selçuk Karabatı

Date: _____

ABSTRACT

Scheduling of Resource Constrained Projects via Genetic Algorithm with Invasive Weed Optimization Reproduction Considering the Multiple Skills of Resources in Multi-Project Environment

Ayşe Bengi Doğan

Master of Science in Industrial Engineering

October 2, 2023

Existing literature on resource constrained project scheduling problems mainly focuses on single-project scheduling problems. However, the research in scheduling multiple projects has been increasing until today. This paper considers a multi-skill resource constrained multi-project scheduling problem (MS-RCMPSP) with global and local resources. Our MS-RCMPSP consists of a multi-project environment with global and local human resources, where their skills and proficiencies of the resources are also considered. We assume that there are precedence relationships between the tasks. The aim is to assign employees to tasks to minimize the completion time of all projects. Employees must be assigned to tasks and the projects by considering their skills and skill levels. Since it has been established that the resource constrained project scheduling problem (RCPSP) is NP-hard, we aimed to obtain the optimal or near-optimal solutions with a heuristic algorithm. We propose a genetic algorithm (GA) with invasive weed optimization (IWO) reproduction component to solve this complex problem. GA enables us to tackle this complicated problem by dividing into task scheduling and resource assignment problems. The IWO reproduction increases the diversity in schedules, which leads to a better solution. A new set of 30 instances with large instances with nearly 500 tasks is generated for MS-RCMPSP with global and local resources. The project known as iMOPSE (Intelligent Multi-Objective Project Scheduling Environment) offers tools that can optimize different schedules using a variety of heuristics or metaheuristics, and we used iMOPSE data generator to create our problem data. Since there is no benchmark data in the literature, the performance of the proposed algorithms was tested on generated 30 instances. We achieved the solution for this unique problem by combining a genetic algorithm (GA) and invasive weed optimization reproduction (IWOR). When we compared it with a classical genetic

algorithm, we concluded that it performed better than the GA in terms of the total makespan for all projects.

ÖZETÇE

Kaynak Kısıtlı Projelerin, Çoklu Proje Ortamında Kaynakların Çoklu Becerileri Göz Önünde Bulundurularak Yabani Ot Optimizasyonu Çoğaltması ile Genetik Algoritma Yoluyla Çizelgelenmesi

Discipulus nomen

Endüstri Mühendisliği, Yüksek Lisans

2 Temmuz 2023

Kaynak kısıtlı proje çizelgeleme problemi üzerine mevcut literatür çoğunlukla tek proje çizelgeleme problemine odaklanmaktadır. Ancak günümüze kadar çoklu proje çizelgeleme ile ilgili araştırmalar artarak devam etmektedir. Bu çalışmada, küresel ve yerel kaynaklarla çok becerili kaynak kısıtlı çok projeli çizelgeleme problemi ele alınmıştır. Bu problem, çoklu proje ortamını, kaynak olarak küresel ve yerel insan gücünü, kaynakların beceri ve uzmanlıklarını ve görev öncelik ilişkilerini içerir. Amacımız, çalışanları tüm projelerin tamamlanma sürelerini en aza indirgeyecek şekilde görevlendirmek olup, çalışanların yetenek ve uzmanlıkları da dikkate alınmıştır. Bu zor problemi çözmek için İstilacı Yabani Ot Optimizasyonu (IWO) ile Genetik Algoritma'yı (GA) birleştiren bir sezgisel algoritmayı öneriyoruz. Genetik algoritma ile eniyi veya eniyiye yakın çözüme ulaşmayı hedeflerken daha iyi bir çözüme ulaşabilmemizi sağlayacak çözüm çeşitliliğini artırmak için IWO çoğaltmayı kullandık. Çok Becerili Kaynak Kısıtlı Çok Projeli Çizelgeleme Problemi (MS-RCMPSP) için, yaklaşık 500 göreve kadar olan büyük örneklere sahip küresel ve yerel kaynaklarla yeni bir 30 örnekli veri seti oluşturduk. Wroclaw Teknoloji Üniversitesi tarafından geliştirilen IMOPSE veri üretici, aslında tek bir proje verisi oluşturmak için kullanılır ancak biz çoklu proje ortamı oluşturmak için bu verileri birleştirdik. Literatürde herhangi bir kıyaslama verisi bulunmadığından, önerilen algoritmaların performansı oluşturulan 30 örnek üzerinde test edilmiştir. Genetik algoritma (GA) ile istilacı ot optimizasyonu çoğaltması (IWOR) birleştirilerek bu benzersiz problemin çözümünü elde ettik. Klasik bir genetik algoritma ile karşılaştırdığımızda, tüm projelerin toplam tamamlanma süresi açısından GA'dan daha iyi performans gösterdiği sonucuna vardık.

ACKNOWLEDGEMENTS

I would like to state my thankfulness and appreciation to the following people for their support in creating this thesis. Firstly, I would like to thank my supervisor, Prof. Ceyda Oğuz, for guiding and supporting me during my masters'. I thank her for guiding me through the complexity of the problem and supporting me with her knowledge during the difficult times I was in. I would also like to thank my family for their unwavering support during these times. I am grateful for their efforts to be by my side during this difficult and stressful process.I want to mention great friends who support me during my masters' degree. Ayşe, Efsun, Oğuzhan and Yağmur, who made me feel lucky to have them in my life, thank you for being there for me every time my motivation is low and I have stressful moments throughout my thesis. I would also like to thank Ayça, the wonderful person I had the chance to meet at work, for motivating me with her messages. The wonderful people I met at Koç University, Çelen, Buse and Kerim, I am very lucky to have received this degree with them. I am especially grateful to Çelen for the support she provided during the thesis process. Finally, I am grateful to my brother Yusuf, who helped me gain my perspective on life. Although we cannot communicate due to his illness, I know that his luck is always with me. Thanks to him, I learned to cope with difficulties and not to give up. I am so lucky to have him in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVATIONS

GA Genetic Algorithm

HGA Hybrid genetic algorithm

IP Integer programming

IT Information technologies

ITS Improved tabu search

IWO Invasive weed optimization

IWOR Invasive weed optimization reproduction

MIG Mixed integer goal programming

MIP Mixed integer linear program

MM-RCPSP Multi-mode resource constrained project scheduling problem

MOGRASP A multi-objective greedy randomized adaptive search procedure

MOIWO Multi-objective invasive weeds optimization algorithm

MOPSO Multi-objective particle swarm optimization algorithm

MS-MMRCPSP Multi-mode multi-skill resource constrained project scheduling problem

MS-RCMPSP Multi-skill resource constrained multi-project scheduling problem

MS-RCPSP Multi-skill resource constrained project scheduling problem

NP Non-deterministic polynomial

NPV Net present value

NSGA-II Non-dominated Sorting Genetic Algorithm II

NV-MOVNS New version of the multi-objective variable neighborhood search algorithm

RCMPSP Resource constrained multi-project scheduling problem

RCPSP Resource constrained project scheduling problem

RCP-SPTT Resource constrained project scheduling problem

with transfer times

RDS Repair-based decoding scheme

TSA Tabu search algorithm

TSP Travelling salesman problem

VNS Variable Neighborhood Search

NOTATIONS

Section 3.2 Notations

Indices

h projects

i, j activities

l, g index of workforce for local and global resources

t time

k index of skills

Parameters

ɳ set of projects

J_h set of task in project $h = \{1,2,3 \dots \dots, H\}$

d_{jh} duration of the task $j_h = \{1_h, 2_h, \dots \dots, j_H\}$in project $h = \{1,2,3 \dots \dots, H\}$

e_{jn} required skill level of S_k for task j_h

q_{ln} the level that resource l proficient for S_k

r_{gn} the level that resource g proficient for S_k

L_h set of local resources in project $h = \{1,2,3 \dots \dots, H\}$

G set of global resources

R set of resources $L_h \cup G$

S set of skills

U set of skill levels

μ_{jh} set of all precedence relationships of task $j_h = \{1_h, 2_h, \dots \dots \dots j_H\}$ in project $h = \{1,2,3 \dots \dots, H\}$

Variables

c_{jh} completion time of the task j in project h

b_{jh} earliest starting time of task j in project h

Figure 3.2 Notations

Variables

s_r the resource early starting time

f_r the resource finishing time

Figure 3.4 Notations

Variables

w represents weed in the IWOR

s_i the number of seeds

Parameters

s_{min} *and* s_{max} are the parameters that define the number seeds generated by the worst and best weeds in the population.

f_{min} *and* f_{max} *represents the minimum and maximum fitness value*

Algorithm 1&2 Notations

Variables

$\boldsymbol{P}_t$ the permanent population

$\boldsymbol{Q}_t$ the newly created population that is created by IWOR

$\boldsymbol{R}_t$ the newly created population that is created by OX2 crossover operator

sch_{opt} optimal solution

Parameters

$maxgen$ maximum number of generation

PopSize population size

p_c crossover probability

p_m mutation probability

Chapter 1:
INTRODUCTION

Project scheduling is one of the most critical managerial issues in organizations. It is frequently encountered in the production, IT, education, and service sector. In most organizations, projects are executed using the same resource pool for different projects and several projects are carried out in parallel. However, when considering today's project management environment, the core problem is more than the execution of multiple projects simultaneously in real-life applications. Another critical point is the uniqueness of the resources that are allocated to projects' tasks. Most resources are considered equal skills and skill levels in the literature, but the situation is different in real project environments.

Another issue effecting project environments is the increasing prevalence of remote working. The pandemic will likely cause an irreversible switch to remote working in most companies. This rise was already seen even before the pandemic, thanks to technological advancements used in companies. Most global companies will be affected by this shift, as they will gain the ability to employ their resources from all around the world. This means that while companies can assign resources to their projects locally, they can also access resources that can support their projects from all around the world. There are several reasons why companies need and prefer this, such as the high wages of the employees in their local area and the need for a team that can support them when they deploy projects in more than one region around the world. In the literature, these resources are called global and local resources. Local resources can only take part in specific projects in specific locations. Global resources, on the other hand, can support all projects in the project pool, independent from the geographies. For example, consider a company that provides services in many countries of the world and let us say they roll out an application to be used in all branches of the company. There are local resources dedicated to that project in every country. Additionally, a global team can remotely contribute to all the projects worldwide. In this case, the company should consider all its local and global resources and they plan its resource planning

accordingly. Considering such factors, the problem in this study extends from the classical problem of scheduling resource-constrained projects (RCPSP). Habibi, Barzinpour, and Sadjadi (2018) define RCPSP as scheduling project activities to optimize an objective function by considering limited resources and precedence relations. According to Ulusoy (2002), the assumptions underlying the definition of the problem can be listed as follows:

- The duration of the task is deterministic.
- The use of resources per unit of tasks is fixed.
- A task can only be executed by a single resource assigned. During the execution, the assigned resource cannot take over another task. Once executed, a completed task cannot be taken over by another resource
- The task initiated must be completed without interruption.
- All tasks planned within the project must be executed. None of them can be ignored.

In addition, according to Ulusoy (2002), it is possible to acquire different definitions of problems by omitting some of these assumptions. The classification of the resource-constrained project scheduling problem is depicted in Figure 1.1, which shows that RCPSP has been divided into many sub-problems so far (Kolisch, Sprecher, and Drexl,1995).

In the literature, resources are classified into three according to their availability

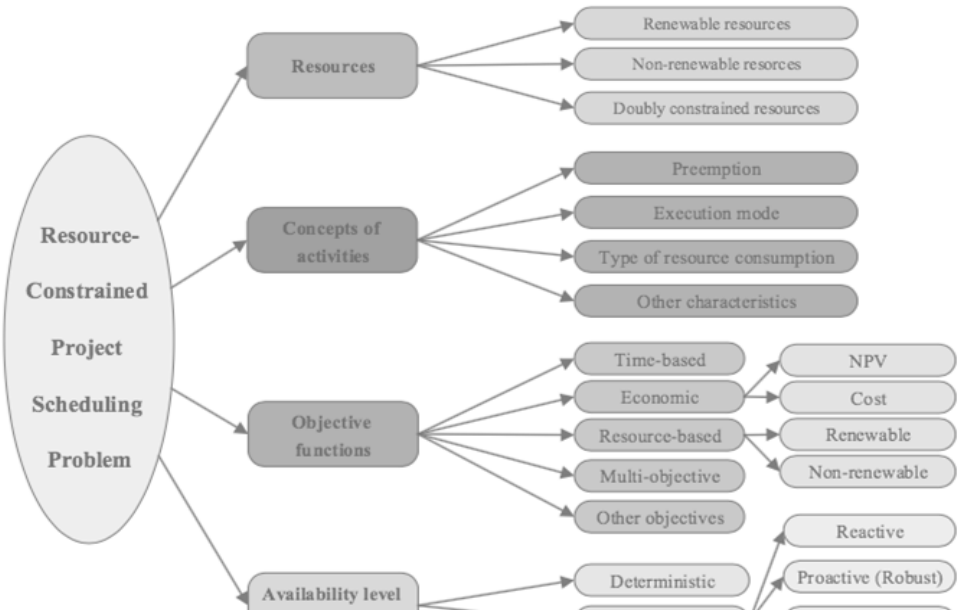


Figure 1.1: Classification of resource constrained project scheduling problem

during the project as renewable, non-renewable, and doubly constrained resources. Renewable resources are available on a period-by-period basis; the available amount is renewed per unit time, e.g., per hour, per day, per week. Only the total capacity of resources at every time unit is constrained. These resources are available once a task is completed; they are not diminished during the execution of the tasks. Machinery, computers, and human resources can be examples of renewable resources. Non-renewable resources, unlike renewable resources, have a limited consumption availability for the entire project and their availability is not updated. Once used for a task, their availability diminishes according to the amount assigned to the completed task. The most common examples of non-renewable resources are money, raw materials, and energy. Doubly constrained resources are constrained periodically and for the entire duration of the project. Their availability is renewed only after the project is completed. The whole project's total budget is an example of doubly constrained resources.

In today's project-based organizations, we observe that the skills and skill levels of the resources, which we take as human resources in this study, significantly impact the execution of tasks in the project. In our problem, human resources, a renewable resource, is discussed and considered that all of the resources have unique characteristics, which means that each resource has different skills and skill levels.

Addressing the highly complex project scheduling problem mentioned above, we expand the RCPSP problem by considering the multi-project environment and the multiple capabilities of resources. Furthermore, considering the remote working system, we also consider global and local renewable resource availability. In other words, in addition to the project teams that provide support to the project locally, we also consider the global renewable resources that can support all of the different ongoing projects in different geographies. Handling all these complexities, we extend RCPSP to MS-RCMPSP, the "Multi-Skilled Resource Constrained Multi-Project Scheduling Problem" with global and local resources. The main motivations behind our extension of the classic RCPSP to MS-RCMPSP with global and local resources are:

- To increase its applicability in real life, as mentioned before, we consider the multi-project environment where projects are run in parallel. Multiple projects are executed at the same time in many project-based working environments. We consider that the skills and skill levels of the resources differ because it is known

that in today's working environments, each human resource has different skills and skills levels. For example, let us consider an IT business analyst and a senior IT business analyst working on the same project: These two cannot have the same skill and skill level when we consider the experience they have gained.

- To consider today's working system, which has changed with the advancement of technology and the pandemic, we can now consider the geographical attributes of the resources in the same projects. No study in the literature deals with the distinction of local and global resources in MS-RCMPSP. Apart from the skill level and skills of the resources, we also consider their geographical conditions and the project types they can support. In fact, with the solution to this problem, the company can also obtain information as to which branch of the company it should assign the global resources to be recruited.

This study aims to minimize projects completion time by assigning resources, i.e. employees, to project tasks according to their skills and expertise. As it has been established that the RCPSP is NP-hard (Blazewicz, Lenstra and Kan, 1983), our problem is also NP-hard solution method is a Genetic Algorithm (GA) with Invasive Weed Optimization (IWO) reproduction. As the solution method in this thesis is based on the principles of genetics, and natural selection is used for solving both constrained and unconstrained optimization. IWO, like GA, is a population-based evolutionary algorithm method inspired by the behavior of weed colonies. GA and IWO are widely used for NP-hard problems to find optimal or near-optimal solutions. They can be used for the NP-hard project scheduling problem and other problems such as Travelling Salesman, job scheduling, and resource assignment problems. Multi-skill resource constrained project scheduling problem (MS-RCPSP), as introduced in this study, is also an NP-hard problem. It is a complicated problem because it is highly constrained in terms of renewable resources. The advantages of GA are its simplicity of adaptation to complicated problems and its suitability for many optimization problems (Yang, 2021). IWO enhances GA in this work by increasing the intensification of the resulting candidate solutions. These advantages are the main motivations behind our choice of this method for solving the MS-RCMPSP to find an optimal or near-optimal solution.

MS-RCMPSP in this study includes a single mode, multi-project, renewable global and local resources with their capabilities. Pre-emption is not allowed, so a task cannot be interrupted during its execution. A resource can only perform one task at a time, and

a task can only be executed by an assigned resource. There are also precedence relationships among tasks. If a task has predecessor tasks, it should be ensured that all the predecessor tasks are completed before that task. GA with IWO reproduction is proposed for MS-RCMPSP in this study, and we generate a new instance set for the problem consisting of 30 benchmark instances with up to 60 to 500 tasks, 2 to 10 projects and 1 to 5 global resources, using the IMOPSE data generator developed by the Wroclaw University of Technology (Myszkowski and Laszczyk, 2022). Since we do not have benchmark data, we evaluate the performance of our algorithm with the generated data.

In summary, we examine a new working model brought about by changing technology and lifestyles in this study. The fact that companies consider different wage policies in different countries or have a high chance of reaching talented people in different global fields has pushed us to choose this problem. Unlike the studies in the literature, the new working model is considered in this study with skills and skills levels of the resources. In addition, the invasive weed optimization algorithm for RCPSP, which is popular today, is compared to GA, GA-IWOR and IWO, and the performance of an algorithm that has newly started to be used in this field is presented to you.

This thesis is organized as follows: Chapter 2 provides an overview of the relevant literature for the problem; GA with IWO reproduction is introduced for solving the MS-RCMPSP in Chapter 3. The computational results of the algorithm and the analysis are presented in Chapter 4. Finally, we summarize the thesis and discuss the future works in Chapter 5.

Chapter 2:
LITERATURE REVIEW

The section presents previous studies related to multi-skill resource-constrained multi-project scheduling problem (MS-RCMPSP) and its sub-problems. This complex problem produces multiple sub-problems, each with its own spectacular set of difficulties. Resource constrained project scheduling problem includes the standard resource-constrained project scheduling problem (RCPSP), the multi-skill resource-constrained project scheduling problem (MS-RCPSP), the resource-constrained multi-project scheduling problem (RCMPSP), and the multi-skill resource project scheduling problem in a multi-project environment. Recently published studies have shown that the MS-RCMPSP is a notably difficult problem to solve because the large number of sub-problems, constraints and variables are involved. To solve this problem, the researchers use different approaches, including heuristic algorithms, meta-heuristic methods, and mathematical programming methods.

2.1 Resource Constrained Project Scheduling Problem (RCPSP)

This section briefly overviews the resource constrained project scheduling problem (RCPSP). Our literature review shows that researchers focus not only to get the optimal or near-optimal solution for classical RCPSP, but also on expanding it by combining different features.

Bartusch (1988) schedules projects by considering starting and finishing times of each job and resource constraints. In addition, precedence relationships are considered when scheduling project. In this study, they choose regular cost functions such as makespan and tardiness cost and select a branch and bound to solve the problem. Hartmann (1998) discusses the resource-constrained project scheduling problem (RCPSP) with makespan minimization as an objective. Two GA approaches, a priority value based and a rule-based priority representation, are proposed to solve this problem. The author emphasizes that it was the most influential genetic algorithm ever obtained. Therefore, this study is essential in using genetic algorithms for the RCPSP.

Cavalcante (2001) also schedules projects by considering labour constraints. They express their study as a simplified version of the industry problems. In this problem, there is a required unit labor force for each job, and the availability of each labor is limited for each period. It is aimed to finish the projects as soon as possible under the precedence and labor constraints. They use the tabu search algorithm (TSA) for the proposed method. In addition, they show that integer programming (IP) gives a fairly good lower bound by changing the objective function. Valls et al. (2008) propose a hybrid genetic algorithm (HGA) for RCPSP. They improve special operators that are applied for generated schedules. They create a newly developed crossover operator for the RCPSP and a local improvement operator for all generated schedules. The proposed algorithm also implements a two-phase strategy to obtain better results In this algorithm, the authors repeat the evolutionary process with the best solution obtained in the first step. Thanks to this algorithm, successful and high-quality solutions have been obtained for some instances in the PCPLIB library. Goncharov and Leonov (2017) also study the RCPSP with makespan minimization as an objective. Resource and technological constraints of activities are considered when scheduling the projects. They suggest two versions of crossover operators with GA to get better solutions. These operators use a heuristic algorithm that considers the necessity of the resources. This study also use the PCLIB library to perform the algorithm. The best solutions are obtained with the proposed method for some of the instances in this library. Kadri and Boctor (2018) introduce a resource-constrained project scheduling problem with transfer times (RCP-SPTT). They aim to find appropriate start times for each activity to minimize project duration. The objective is to minimize the makespan by considering precedence relationships, resource capacities, and resource-transfer time constraints. GA is proposed as a solution method for the problem and implemented a two-point crossover operator. The authors state that they get better results than some previously published results.

2.2 Multi-skill Resource Constrained Project Scheduling Problem (MS-RCPSP)

As an extension of classical RCPSP, Bellenguez and Neron (2005) introduce multi-skill resource constrained project scheduling problem (MS-RCPSP). They consider that the resources have different skills and skill levels in their problems. In other words,

while scheduling the project, they assign their resources, considering the needs of the tasks for these skills and capabilities. They propose two lower bounds to minimize the project makespan. Another similar work is conducted by Kadro and Najid (2006) and they extend a TS (Tabu Search) solution method combining by the solid local search for multi-mode resources constrained project scheduling problem (MM-RCPSP). They classify workers into three levels, senior, standard, and junior. Gürbüz (2010) also introduce MS-RCPSP with hierarchical levels of skills. Different abilities and skill levels have been determined for each resource when scheduling project tasks in this problem. Different from other studies, two objectives are selected: minimizing the completion time of the project and the total skills wasted. Gürbüz (2010) propose a multi-objective genetic algorithm to minimize both objectives.

Similarly, Yannibelli and Amandi (2013) conduct a study that gives detailed understanding for multi-objective MS-RCPSP. They consider resources, skills and skill levels when scheduling the project tasks. They use hybridizing a multi-objective simulated annealing algorithm and genetic algorithm to minimize project makespan and maximize the effectiveness levels of the employees.

Maghsoudlou and Sadeghzade (2016) introduce a multi-skill multi-mode resource constrained project scheduling problem (MS-MMRCPSP) with three objectives:

- minimizing the makespan of the project,
- minimizing the total cost of allocating workers to skill, and
- maximizing the total quality of processing activities.

For this problem, a multi-objective invasive weeds optimization algorithm (MOIWO) with a new chromosome structure that ensures the feasibility of the schedules is proposed. Two meta-heuristic algorithms, the non-dominated sorting genetic algorithm (NSGA-II) and the multi-objective particle swarm optimization algorithm (MOPSO), are used to evaluate the proposed solution. Experimental Results indicate that the MOIWO algorithm performs better. However, regarding the diversification metric, the MOPSO algorithm exhibits better performance, whereas the NSGA-II algorithm demonstrates better performance in terms of the spread of the non-dominant solution. Dai et al. (2018) consider the MS-RCPSP with step function of processing time of each task and deteriorating date. According to the problem's features, four neighborhood structures and two mutation operators are developed to improve the tabu search algorithm (TS). They measure the algorithm's performance using the test instance from

the literature. The proposed method provides a better solution than the best-known results regarding solution quality. According to Lin, Zhu and Gao (2019), MS-RCPSP can be generalized into two sub-problems: task scheduling problem and resource assigning problem. They propose a genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem. Simple heuristic rules are used to construct a set of low-level heuristics. To enhance these low-level heuristics Genetic programming is employed as the high-level strategy; a simple encoding scheme and repair-based decoding scheme are developed to obtain feasible schedules in an efficient way. They measure the performance of a new genetic programming hyper-heuristic approach using the IMOPSE library. In this thesis, we also use IMOPSE data generator to create random test instances.

2.3 Resource Constrained Project Scheduling Problem in Multi-project Environment (RCMPSP)

In this section, we examine the resource constrained multi-project scheduling problem (RCMPSP) to consider the environment where multiple projects are carried out at the same time. According to de Boer (1998), a single-project problem can be quickly transformed into a multi-project problem. Suppose we deal with H projects, where each project (h=1,……, H) contains a number of activities. This multi-project environment can be converted into a single project problem by joining these H projects into one new mega project. Later, Besikci, Bilge, and Ulusoy, (2012) conduct a study which shows two main approaches to scheduling multiple resource-constrained projects: The first approach is that each project is handled on its own and the second one has integrated all projects into one mega project. In other words, In the first approach, we assume that each project has its own start and end node. In the second approach, we assume that there is a common start and end node for all projects. They use the second approach to solving the Resource portfolio problem under a relaxed resource dedication policy in a multi-project environment. They propose a Genetic Algorithm (GA) to reach an optimal or near-optimal solution.

Gonçalves et al. (2008) study RCMPSP with minimizing finishing time for all projects. They propose GA with the random key representation of chromosomes. The activities are scheduled by considering two constraints: (1) precedence relationships

between the activities, and (2) the capacity of the resource assigned to the activity, which should be enough to complete it. Since there is no benchmark data in the literature, they perform the experiments by generating random test instances. The calculated results show that the proposed algorithm is effective in solving RCMPSP. Suresh et al. (2015) also proposes a new GA to solve the RCMPSP. They extend the problem with resource transfer times, in which the resources are considered in different geographic locations. The time and cost of resources when they change the location for projects are considered. A new GA has maximized all projects' net present value (NPV). Two-phase priority rules are also implemented for the same problem. The paper presents extensive experimental results. They use 60 two-phase priority rules to analyses and compare them with the GA approach.

2.4 Multi-skill Resource Constrained Multi-project Scheduling Problem (MS-RCMPSP)

In this section, we consider the problem of multi-skill resource constrained multi-project scheduling, which is the main topic of this study. There are few studies on this subject in the literature, but the interest is increasing day by day. Due to its proximity to real-life data, researchers focus on this subject, and many papers have been published recently. Heimerl and Kolisch (2009) introduce scheduling and allocating multiple projects with a multi-skilled workforce. They consider internal and external human resources to assign IT projects. Contrary to other studies, they assume that the schedule but not the start of each project is given. They only consider the resource assignment problem. The objective is to minimize worker costs because the employees' average wages are different from each other. The researchers develop a mixed-integer linear program (MIP) and present the advantages of using it over employing simple heuristics to achieve feasible and cost-effective solutions. Walter and Zimmermann (2016) introduce a staffing problem with multi-skill and multi-project environments. They assign employees to the projects by considering the skill requirements of each project. The objective is to minimize the average team size of projects because large teams often face challenges in coordination and experience a loss of motivation due to frequent project switches. They develop a mixed-integer linear program and, for large instances, and formulate three heuristics. They generate their own instances based on a study of

real-life instances provided by a project partner company. Chen et al. (2017) addresses a multi-objective model for multi-project scheduling and multi-skilled staff assignment for IT product development, considering competency evolution. The product development environment within IT has been considered when creating the problem. Product development projects take place in parallel and include several scheduled activities using the proposed algorithm. When assigning the resources, the improvement of the capabilities and skills of the resources through learning processes is considered. Three objective functions are chosen according to the demand of product owners or managers based on real life: (1) skill efficiency gain, (2) product development cycle time, and (3) costs. The authors propose a multi-objective non-linear mixed integer-programming model to solve this complex problem. The data is obtained from a leading electrical device company in China to perform experiments. Hosseinian and Baradaran (2021) design a two-phase approach based on Statistical Process Control (SPC) to solve a multi-objective MS-RCMPSP. A Multi-Objective Greedy Randomized Adaptive Search Procedure (MOGRASP) is used in the first phase. In the second phase, the MS-RCMPSP is solved via a new version of the multi-objective variable neighborhood search algorithm (NV-MOVNS). The paper examines MS-RCMPSP by minimizing the completion times and costs of all projects. This bi-objective problem is solved by considering four crucial aspects: (1) familiarity with resources determines the length of activities, (2) the more productive a worker is, the higher her/his salary, (3) projects have different due dates, and (4) the budget of each period varies over time. Since there is no benchmark data in the literature, instances obtained from a construction company in Iran are used to perform the proposed algorithm. The results indicate that a two-phase statistical process control (SPC) approach can be implemented in real-world multi-project scheduling problems. Cui et al. (2021) addresses a multi-mode and multi-skill resource constrained multi-project scheduling problem (MM-MSRCMPSP) to solve their partner company problem. They extend MS-RCMPSP to multi-mode MS-RCMPSP. The objective is to minimize the total makespan with multi-skilled resources that have different skill levels and differences in processing times for the same task can arise. They propose Variable Neighborhood Search (VNS) to solve the problem. Haroun et al. (2022) also consider MS-RCMPSP. The difference in their studies is that the activities are preemptive with release and due dates, and there are no precedence relationships among the tasks. The paper aims to minimize the total weighted tardiness

and the deviations of undesirable goals by completing all the tasks within the planned project deadline. They design a mixed–integer goal programming (MIGP) to obtain optimal or near-optimal results. Since there is no benchmark data in the literature, they used data based on real-life.

Chapter 3:
METHODOLOGY

In this chapter, we provide the problem description, the mathematical model representation of the problem, and the solution methodology. We discuss the motivation behind the problem and the formal description of the problem in Section 3.1. The mathematical model representation of the problem and a Genetic Algorithm (GA) with Invasive Weed Optimization (IWO) reproduction as a proposed solution is presented in the following sections 3.2 and 3.3, respectively.

3.1 Problem Definition

In our problem, multiple projects must be scheduled simultaneously. Tasks within the projects must be scheduled by minimizing the makespan of the whole projects, and appropriate resources must be assigned to tasks. We have limitations on resources and tasks when solving this problem; particular skills and skill levels are needed for each task, and at the same time, the precedence relationships between these tasks must be considered. The skill level of a resource must be equal to or greater than the required level of the task to which it will be assigned. Also, the skill levels are not additive. This means that two or more resources cannot be used to obtain a skill level for a specific task. A resource can only complete one task at a time and a task can only be executed by one resource. At the same time, once a task is started, this task cannot be interrupted; that is, tasks are non-preemptive. In addition, unlike the MS-RCMPSP in the literature, in this problem, the resources are divided into global and local. While local resources can support projects in their region, global resources can support any project and we named this problem as multi-skill resource constrained project scheduling problem (MS-RCMPSP) with global and local resources.

To illustrate this problem, let's assume that we are in a global company and imagine a team that provides support for the roll-out of digital products owned by the company. There are resources in every country for the roll-out of these digital products, and there are employees at the headquarters who support all these countries, which are global

resources. However, the competence and competency of the resources in these teams should also be considered because a team may include a junior employee, a senior employee and even a project manager. In addition to all these constraints, the precedence constraints of the projects' tasks are also considered. A task with preceding tasks can only be executed if these predecessors are completed. The following sections provide a more formal and detailed problem definition and the notations used in the formal definition are presented in Table 3.1.

Table 3.1: Notations are used for the problem definition

The notations
Indices
h projects
i, j activities
l, g index of workforce for local and global resources
t time
k index of skills
Parameters
ɳ set of projects
J_h set of task in project $h = \{1,2,3 \dots \dots, H\}$
d_{jh} duration of the task $j_h = \{1_h, 2_h, \dots \dots, j_H\}$ in project $h = \{1,2,3 \dots \dots, H\}$
e_{jn} required skill level of S_k for task j_h
q_{ln} the level that resource l proficient for S_k
r_{gn} the level that resource g proficient for S_k
L_h set of local resources in project $h = \{1,2,3 \dots \dots, H\}$
G set of global resources
R set of resources $L_h \cup G$
S set of skills
U set of skill levels
μ_{jh} set of all precedence relationships of task $j_h = \{1_h, 2_h, \dots \dots, j_H\}$ in project $h = \{1,2,3 \dots \dots, H\}$
Variables
c_{jh} completion time of the task j_h in project h
b_{jh} earliest starting time of task j_h in project h

3.1.1 Projects and tasks

If we formally present the problem description, we need to schedule H projects which belongs to set η, where each project is identified by their index $h = \{1,2,3 \dots \dots, H\}$. Each project consists of a set of non-preemptive tasks J_h in project $h = \{1,2,3 \dots \dots, H\}$. The last task and first task are dummy tasks, that are used for creating projects network, have zero duration and no resource needed. The tasks $j_h \in \{1, \dots \dots, |J_h|_h\}$ of project h, are scheduled by considering precedence relationships μ_{jh} among the tasks. If task $j_h \in J_h$ has predecessors' tasks i_h, it should be ensured that all the predecessor tasks must be completed before the task itself. Lastly, for each task j_h, where h indicates project h, a starting time b_{jh} must be calculated.

3.1.2 Local and Global Renewable Resources

The tasks j_h are executed by resources from global renewable resource set G and local renewable resource set L_h. As per the definition given in previous section, renewable resources are available on a period-by-period basis; the amount that is available is renewed per unit time, such as per hour, per day, per week, or per activity. While each global resource $g \in G$ can be used in all H projects, each local resource $l \in L_h$ can be used in project h, which is the project conducted in their local geography. In addition, each task j_h should be executed by a single resource and a resource can be assigned to a single resource at a time. This implies that that there is no consideration for resource capacity, like 8-hour daily capacity of a resource being shared by several tasks.

3.1.3 Skills and Skill Levels

In the project h, there are $|S|$ number of skills needed to execute its tasks. Each global resource $g \in G$ and local resource $l \in L_h$ have S set of skills and $|U|$ number of skill levels $u = \{1,2,3, \dots, U\}$. Each resource has several skills at different levels. Each task j_h needs a minimum level skill S_k of a specific type. Every global $g \in G$ and local $l \in L_h$ resources can be assigned to task j_h as long as its skill S_k meets the minimum skill levels requirement for taskj_h. In this problem, skill levels are not

additive. This means that two or more resources cannot be used to obtain a skill level for a task.

3.1.4 Objective

The objective is to minimize the makespan of all projects $max\{c_{jh}, j_h \in J_h, h \in H\}$,where c_{jh} is the completion time of the task j_h in project h, by considering following constraints : (1) Each local resource $l \in L_h$ can be assigned to the specified local project h, (2) Each global $g \in G$ and local $l \in L_h$ resource should be assigned to a task as long as its skill and skill levels are sufficient for the task j_h ,and (3) If task j_h has predecessor tasks i_h, it should be ensured that all the predecessor tasks must be completed before the task itself.

3.2 Proposed Algorithm for the Problem

In this section, a genetic algorithm with invasive weed optimization is presented for solving resource constrained multi-project scheduling problem that involve global and local resources with multi-skills. We illustrate the implementation of the proposed genetic algorithm by introducing the solution representations and shows all the algorithms and their components that comprise the proposed algorithm. Besides, you can find the standard GA and invasive weed optimization algorithm that we use to evaluate the performance of our proposed GA-IWOR algorithm.

3.2.1 Encoding and Decoding the Solution

Determining the best structure to represent the solution is one of the most critical decisions to be made while creating the GA. The representation of the chromosomes directly affects mutation and crossover operators' performance. Finding an efficient representation makes the search easier by restricting the search space; similarly, deficient representations can cause a more expanded search space. We select the permutation representation of the chromosomes. In this structure, every chromosome is a string of numbers representing a sequence task. We represent genes as strings because we aim to get an ordered list of tasks, resulting in the minimum completion time of all the projects in the solution.

3	2	1	4	7	6	5	8	9	10

Figure 3.1: Solution representation of the MS-RCMPSP

MS-RCMPSP includes two sub-problems, which are the task scheduling problem and the resource assignment problem. Hence, we need to design all the solution representation by considering two sub-problems and all the tasks as task list according to the constraints of the problem must be included in the encoding system. At the same time, we need to quickly determine resource assignments in the solution data structure. For example, in Figure 3.1, two projects are merged into one. The first project has the tasks 1, 2, 3, 4, 5, 6, whereas the second project has the tasks 1, 2, 3 and 4 and we re-number them as if they were tasks that belonged to a single project. Thus, tasks 1, 2, 3 and 4 in the second project become tasks 7, 8, 9 and 10, respectively.

3.2.2 Creating Multiple Projects Environment

As the first step of extending the single project scheduling problem to multi-project scheduling problem, Besikci, Bilge, and Ulusoy, (2012) present us two main approaches to scheduling resource-constrained multiple projects.

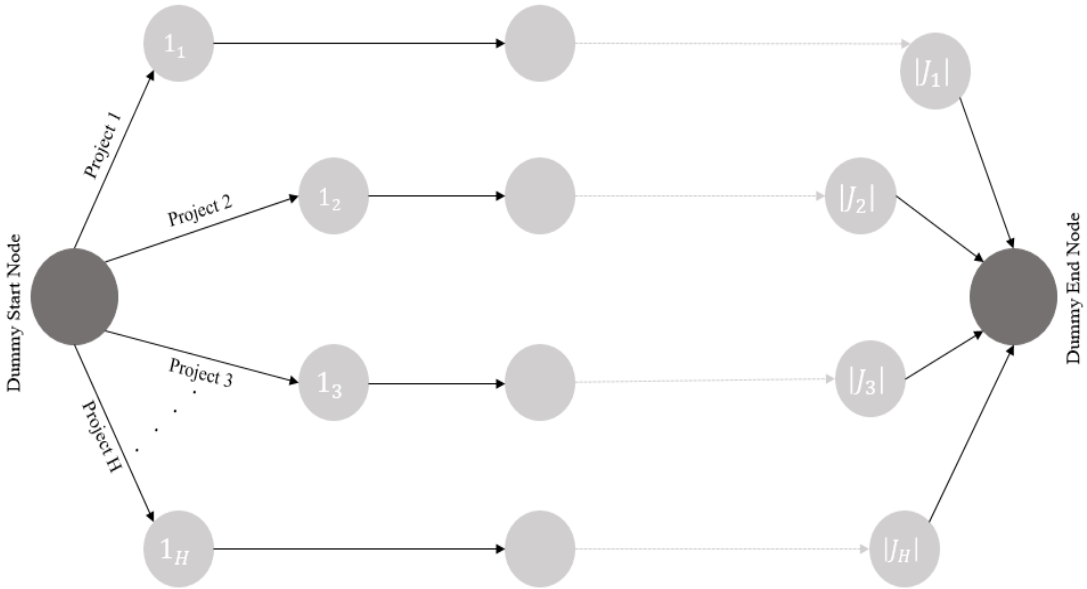


Figure 3.2: Integration of multiple projects into a single project

The first approach is where each project is handled and executed on its own. In other words, each project has its starting and ending node, and the second one is where all projects are integrated into one mega project. In other words, the start and an end node is the same for all projects in the second approach. Similarly, in Figure 3.3 we select to combine each single project into one large project because we consider projects that are executed simultaneously. Boer (1998) indicates that the dummy start and the dummy end nodes help us to transform each single project to one mega project. Suppose we deal with H projects, where each project h ($h = 1, \dots \dots, H$) contains J_h set of tasks. These projects that are conducted in parallel can be converted into a single project by joining start node of each project h to a dummy start node and the end node of each project h to a dummy end node.

Tasks	**Required Skills**	**Duration**	**Predecessors**	**Project**
1	S_1=1	3		1
2	S_2=3	5		1
3	S_3=1	2	5	1
4	S_3=2	4		1
5	S_0=1	6		1
6	S_2=1	5		1
7	S_0=3	3	9,10	2
8	S_1=2	4		2
9	S_3=1	8		2
10	S_0=2	2		2

Table 3.2: Resource-skill relationships

Resources	**Skill Levels**	**Project**	**Type**
1	S_1=2, S_2=1	1	local
2	S_2=3, S_3=2	1	local
3	S_3=3,S_1=2	2	local
4	S_0=3	2	local
5	S_0=3,S_2=2	0	global

Table 3.3: Task-skill relationships

3.2.3 Auxiliary Algorithms and Components

In this section, we go into the algorithms used to generate feasible schedules and elaborate on the strategic implementation of a skill matrix. By ensuring accurate resource allocation to each task, and this integration maximizes our operational effectiveness.

Feasibility Function

As stated in the previous sections, the objective is to minimize the total makespan of the projects. At the same time, each local resource l_h must be assigned to a specified project h and each global resource g and local resource l_h should be assigned to tasks if their skills and skill levels are sufficient. Besides, the precedence relationships between tasks must be ensured. A repair-based decoding scheme (RDS) with left-shift approach is implemented to ensure all the constraints. The idea of the novel repair-based decoding scheme is proposed in Lin, Zhu and Gao (2019), and used to build feasible solutions for the problem. They state that RDS limits the search space uniquely for reaching feasible solutions efficiently.

The example schedule can be seen in Figure 3.2. Here it is seen that task 3 is ahead of its predecessor, task 5. These precedence relationships can be checked in Table 3.2 Task-skill relationships. The repair-based decoding scheme's algorithm, you can find pseudo-code in Figure 3.3, also ensures precedence relationships between the tasks. First, we need to check whether there is a predecessor task and whether it takes place before its successors. If the precedence requirement is not met, it places at the end of the schedule, thus guaranteeing that the predecessor task will be executed before its successors; the process continues until all precedence relations are checked. Returning to task 3, since task 3 takes place before task 5, task 3 is placed at the end of the schedule here. RDS is supported with this method to provide a precedence relationship.

Figure 3.3: Pseudo code for repair-based decoding scheme

Initialize tasks' durations ($d_{jh} = 0 \ \forall \ j = 1, \ldots . . J, h =, \ldots \ldots . . H$)

Initialize the tasks starting times to zero ($b_{jh} = 0 \ \forall j = 1, \ldots \ldots , J \ , h = 1, \ldots . . H$)

Initialize makespan of the tasks to zero ($c_{jh} = 0 \ \forall j = 1, \ldots . . , J \ , h = 1, \ldots ..$H)

Next For each project h=1,…….,H

For the number of each task $|J_h|$ j=1,…..,J , h=1,……,H

For each resource R^r r=1,……..,R

If all predecessors of the task j_h are processed

Compute the start time of the task on each resource by the left-shift scheme (1):

b_{jh}=max{s_r, max ($b_{ih} + d_{ih}$)} where s_r represents resource early starting time. If the task has no predecessors' tasks, then max($b_{ih} + d_{ih}$) is equal to 0 and the earliest start time of the task q_{jh} equals to resource early starting time s_r.

Else

Assign the task to the end of the schedule.

End

End

End

(1) RDS (Wang and Zheng, 2018; Zheng et al., 2017) uses a left-shift approach for local search, improving resource utilization by compactly transferring tasks to the left.

Lin, Zhu and Gao (2019) also utilized the left shift method, an effective local search technique, for RDS to process tasks more efficiently. They aim to decrease the idle time between tasks on a resource due to task precedence constraints. The assignment procedure of resources to tasks is made from left to right in schedule, and it is ensured that the tasks are scheduled at the beginning of the schedule are assigned to one of the resources before. The left-shift method provides better and more effective feasible solutions by reducing the idle time. The starting time of the task on each resource is calculated by the left-shift method as follows: Let us take the solution in Figure 3.2 again. Task 3 is placed at the end of the schedule because it has task 5 as a predecessor, which means that task 5 must be executed before task 3. The earliest starting time of task 3 is calculated as formula 3.1:

$$b_{31}=\max\{s_r, \max (b_{51} + d_{51})\} \quad (3.1)$$

Here, b_{31} represent the earliest starting time of task 3 and s_r represents the resource early starting time, which means earliest available time for the resource r. In the formula, the earliest starting time of task 3 equals to the maximum of the resource early starting time or completion time of the predecessor task 5. If a task j_h has no predecessors, then $\max(b_{ih} + d_{ih})$ is equal to 0 and the earliest start time of task j_h equals to resource early starting time s_r.

Another issue is that resources may be idle while the preceding task is waiting to be completed. In this case, we must check the interval of this idle time $[s_r, f_r]$, where s_r represents the earliest starting time for the resource and f_r is the completion time for resources. If a task can be completed within this time interval $[s_r, f_r]$, that task can be scheduled to this interval. Then, the earliest available starting time is calculated with formula 3.2:

$$b_{jh}=\max\{s_r, \max(b_{ih} + d_{ih})\} + d_j \leq f_r \quad (3.2)$$

The Skill Matrix

Lin et al. (2019) proposes a resource-task assignment matrix for RDS with left-scheme method. This matrix enables the allocation of suitable resources to tasks based on their skill levels. Global resources meeting the skill requirements can be assigned to tasks in any project, while local resources can only be assigned to tasks within their geographic area, regardless of their qualifications for projects in other locations.

Tasks/ Resources	1	2	3	4	5
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
	can be assign		cannot be assign		

Table 3.4: The constructed skill matrix

Let us take resource 1 in Table 3.2 as an example; it has skills 1 and 2 at skill levels 2 and 1, respectively. When we examine all the tasks in project 1 from Table 3.3 local resource 1 has sufficient skills and skill levels for task 1 and task 6 so we can assign resource 1 to tasks 1 and 6. Table 3.4 shows these tasks are assigned to resource 1 in the skill matrix and other assignments.

We want to point out that the skill matrix does not represent the resource-task assignment, which represents the solution. It only shows which resources can execute which tasks as per the sufficiency of their skills and skill levels. While performing the RDS with the left-shift scheme method, the task-resource assignment matrix is used to obtain a feasible assignment while searching for the resource that can perform the task in the resource pool. The task-resource assignment matrix also allows us to have a quick search by restricting the resource pool.

3.2.4 Standard Genetic Algorithm

The genetic algorithm (GA) is a metaheuristic algorithm inspired by natural selection. Genetic algorithms are often applied to solve optimization problems that cannot be solved in realistic times. John Holland and his collaborators developed GA in the 1960s and 1970s. Holland and his collaborators were inspired by biological evolution based on Charles Darwin's (1859) theory of natural selection. This biological evolution process was appropriately encoded in the algorithm and is used for real-life problems ever since.

A population of successor solutions evolve towards better solutions in genetic algorithm. Chromosomes represent all the candidate solutions so that different genetic operators can be performed. Mutation and crossover operators are used to increase diversification and intensification in the population. Individuals selected for reproduction from the population are called parents in GA. They produce new individuals, which is called offspring in GA, by the help of reproduction operators. The evolution process of the solutions starts with the generation generated with the specified heuristic rules or randomly generated. In the algorithm, new generations are created iteratively, and some individuals with promising results are selected and passed on to the next generation. A selection operator is used when selecting individuals giving these promising results. Selection is made according to the fitness function results, which is

usually the objective function for the problem. The algorithm concludes either by achieving the allowed maximum number of populations or satisfying the stopping criterion. GA is frequently used in timetabling and scheduling problems in the literature. The use of GA in scheduling was examined in detail in Portmann's survey (1996). It has been emphasized that GA has advantages in scheduling problems. According to Yang (2021), the advantages of genetic algorithms are the simplicity of adaptation to complicated problems and the fact that it is frequently used in many optimization problems. Similarly, Lim et al., (2017) state that the genetic algorithm is usually used for machine learning, optimization problems, and many other kinds of applications. In addition, as Ozdamar (1999) states that Instead of a single point, the population is searched for the best result, and probabilistic rules are used instead of deterministic ones, allowing flexibility in finding the best solution.

Initial Population

Population initialization is the first step in the GA process. Population $\boldsymbol{P_t}$ includes a set of chromosomes that represent the solutions. The initial population $\boldsymbol{P_0}$ is the first generation for the algorithm, is derived populations $\boldsymbol{P_t}$ at generation t={1,2,3, … . . }. Each population $\boldsymbol{P_t}$ has two aspects: the population size $PopSize$ and the chromosome size (which is equal to number of tasks). Creating an initial population is essential for more qualified, diversified generations in the genetic algorithm (GA).

As a first step of initializing the population, we generate an individual randomly.

Selection

The GA selection process is used to decide which individuals in the population will be passed on to the next generation to create offspring. This selection is made according to the result of the fitness function. We mentioned earlier that the fitness function generally equals to the objective function. In this thesis, the fitness function is also equal to the objective function and the fitness function value is calculated with formula 3.3:

$$v_i = f\left(max\{c_{jh}\ , j \in J_h, h \in H\ \}\right), \tag{3.3}$$

Where v_i is the fitness function of individual i_h, and c_{jh} represents completion time of

task $j = \{1,2,3, \ldots \ldots |J_h|_h\}$ and project h=$\{1,2,3,4 \ldots . . H\}$. In this thesis, we used the rank selection method to decide which individuals join the next generation. This method is usually used in problems where fitness values are close to each other. The selection of the individuals depends on the ranks, which is determined according to their fitness value. We keep the $PopSize$ number of the individuals with the best fitness value and remove the remaining individuals from the population. Individuals to be passed on to the next generation are selected as follows: the new population created with GA operators and IWOR includes all the individuals in the population. The best individuals from this new population are selected to form the permanent population. The best individuals are chosen for the new population by rank selection method and transferred to the next generation.

Crossover Operator

As stated in Kora and Yadlapalli (2017), a simple definition for the crossover operator is that the genes that are selected from parents with better fitness value and creates new offspring. There are various types of crossover and mutation operators, and selecting the appropriate operators is one crucial point affecting algorithm efficiency and quality. Umbarkar and Sheth (2015) propose to classify crossover operators into three categories: (1) classical standard crossover operators, (2) binary crossover operators, and (3) application-dependent crossover operators. Each operator differs in terms of usage, advantages, and disadvantages. This thesis uses the one of the application-dependent crossover operators called order-based OX2 operator.

In section *Gene Encoding*, we state that our gene representation is selected as permutation representation. Genes that represent the tasks in the projects with an appropriate order form each chromosome. In this structure, each task and project are represented as strings. Also, the problem we propose is a project scheduling problem, which has led us to choose an order-based crossover operator. The order-based operator (OX2), mainly suggested for permutation encoding (Umbarkar and Sheth, 2015), is used as a crossover operator in the proposed genetic algorithm. While the OX2 has more use in travelling salesman problems (TSP), it is also recommended for scheduling problems (Umbarkar and Sheth, 2015). In the permutation encoding, Coric et al. (2017) uses OX2 order-based crossover operator in their GA. Viana et al. (2020) also uses OX2 operator to guarantee feasibility in crossover process to schedule job-shop tasks.

The OX2 operator is designed to prevent sub-tour in TSP problems. This is to ensure that the method only matches to the order in which the city appears in the parents. Elimination of the sub-tours can also be used for the project scheduling and scheduling problems. Just as we only want the salesman to visit the same city once in TSP, we do not want to execute the same task twice in project scheduling problems. Using the study of Umbarkar and Sheth (2015), we can summarize how the operator is formulated as follows. The OX2 operator randomly selects various positions in a parent chromosome, and the order of the genes in the selected positions of this parent is imposed on the other parent. Let us assume that two individuals of the current population have been selected for crossover. In Figure 3.4, we have two individuals:

$Parent_1$=< 8 2 1 4 9 6 5 3 10 7 > and $Parent_2$ =< 5 2 9 10 7 6 3 8 1 4 >

Initially, some positions are selected from one of the individuals randomly and suppose that third, fifth and seventh positions are selected. The tasks in these positions at $Parent_2$ are 9, 7 and 3 respectively. In $Parent_1$, tasks 9, 7 and 3 are located at the fifth, tenth and eighth positions. Then, $Parent_1$ is copied to create the $Offspring_1$, leaving the third, fifth and seventh positions empty and $Offspring_1$ is equal to < 8 2 1 4 ∗ 6 5 ∗ 10 ∗> . We add the missing tasks to the $Offspring_1$ in the same order in which they appear in the second parent. This results in < 8 2 1 4 9 6 5 7 10 3 >. The same procedure is also applied for the second offspring, exchanging the role of the first parent and the second parent, using the same selected positions. By this method, we create $Offspring_2$ < 1 2 9 10 7 6 3 8 5 4 >.

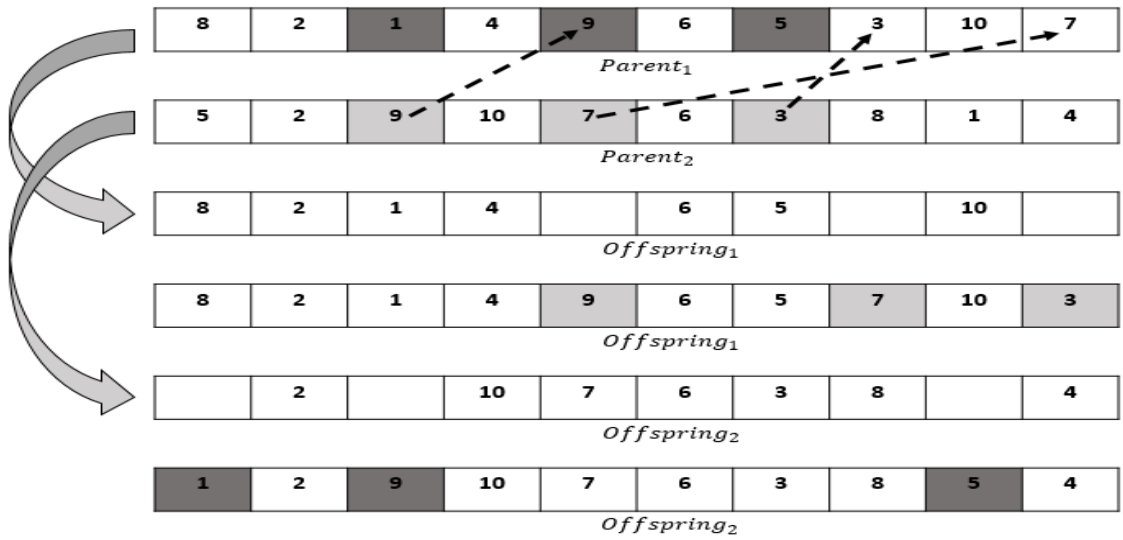


Figure 3.4: The OX2 operator procedure

Mutation

Mutation creates an offspring by randomly changing parent's characteristics. The mutation operator is crucial in genetic algorithms (GAs) as it plays a significant role in preserving variety within the population to avoid early convergence. This method helps to maintain diversity in a population and allows a way to avoid being stuck in local optimal solutions. After performing crossover to create an offspring, the mutation operator is applied separately to each offspring with a probability of p_m.

As stated in Chieng and Wahid (2014), there are several types of mutation operators such as flipping mutation, interchanging mutation, boundary mutation, and reversing mutation.

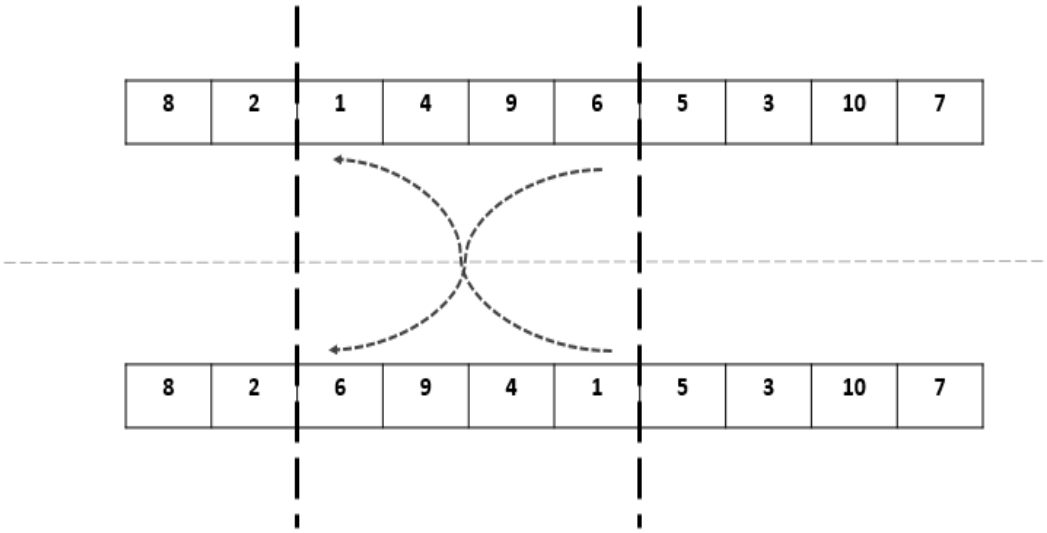


Figure 3.5: The Inversion Mutation Operator

In the proposed algorithm, we select the inversion mutation operator that first proposed by Holland (1975) in his genetic algorithm. It has been used in many studies that have been developed until today. In general, two positions in the chromosome are selected randomly to swap tasks in the selected range. Let's take $Offspring_1$ as an example in Figure 3.5. The range from the 3rd to the 7th gene is selected $Offspring_1$. To ensure the precedence relationship between genes, the feasibility function controls each offspring after crossover and mutation operators are applied.

Algorithm 1: *Standard Genetic Algorithm (GA)*

1. Set parameters: population size (PopSize), maximum number of generation (*maxgen*), crossover and mutation probability (p_c, p_m).
2. Generate an initial population $\boldsymbol{P_0}$ with PopSize individuals. Chromosomes are generated randomly.
3. **Apply** feasilibility function.
4. At the beginning, sch_{opt} is set as empty and its completion time is assigned as infinite.
5. **For** [t=1,$maxgen$]

   5.1 Select pairs of individuals (parents/weeds) by using roulette wheel.

   5.2 Produce offsprings by applying:

   5.1.1 The OX2 crossover and mutation operators are applied to each pair of parents with a probability p_c and p_m.

   5.1.2 **Apply** feasibility function to obtain feasible offsprings.

   5.1.3 **Evaluate** the fitness of offsprings.

   5.3 **If** the best individual's completion time from the entire population is shorter than the completion time of sch_{opt}, then sch_{opt} is updated with this individual's record.

   5.4 Add the children to the population.

   5.5 Reduce population to $PopSize$ by rank selection.

   end for
6. **Return** the best solution found sch_{opt} so far.

The genetic algorithm's parameters are settled at the start, such as population size (PopSize), maximum number of generations (*maxgen*), crossover probability (p_c), and mutation probability (p_m). The initial population is generated randomly to create the chromosomes. To obtain feasible solutions, feasibility function is applied to randomly generated population. The fitness of each schedule is determined by the completion time calculated through the fitness function. The roulette wheel operator is used for selecting pairs of individuals to produce offspring. The OX2 crossover and mutation operators are applied to each pair of parents with a probability p_c and p_m. Again, the feasibility function is applied to ensure that the produced offspring give

feasible results. After all those steps, the offspring and parents are sorted based on a preset fitness value, and those with good fitness move onto the next generation to continue to reproduce. The algorithm keeps running until $maxgen$, the number of iterations, is reached.

3.2.5 Invasive Weed Optimization (IWO)

The invasive weed optimization (IWO), developed by Mehrabian and Lucas (2006), is a population-based meta-heuristic. It is inspired from the way invasive weeds spread and colonize in the natural world. As stated in Cai, Chang and Zhang (2020), there are not a lot of studies on IWO's use in RCPSP, despite the fact that it has been successfully employed in numerous areas. As it is also seen in our literature review, Maghsoudlou and Sadeghzade (2016) use improved IWO in their RCPSP firstly. They also prove that invasive weed optimization (IWO) shows better performance in terms of the diversification metric other than the multi-objective particle swarm optimization algorithm (MOPSO) and non-dominated Sorting Genetic Algorithm II (NSGA-II). Then, Cai, Chang and Zhang (2020) propose to use improved IWO for the standard RCPSP.

The IWO method pursues solutions iteratively using five sequential operations: (1) initiation, (2) reproduction, (3) spatial dispersal, (4) competitive exclusion, and (5) evaluation of fitness of each individual. The IWO, as a population-based algorithm, has some identical specs such as creating offspring based on parents' fitness, increasing population size, and generating new population using the best parents and offspring. Depending on the results of the fitness function, which is the objective function in the algorithm, different numbers of plants can be produced from one weed. In other word, how many new plants will be formed from a weed is calculated with the number of seeds. The weed that is get to a better solution; the number of seeds that belongs to it becoming higher. For example, for our problem, the schedules are called weeds in the IWO algorithm that give minimum makespans; it will have more seeds and produce more plants. One of the strengths of this algorithm is that the better the solutions can reproduce more new solutions. In the following sections, how the number of seeds is calculated, and new solutions are produced are explained in detail.

Since only the invasive weed optimization reproduction is used to improve the performance of the GA in this study, the reproduction process is described in more detail below:

- Initialization

We select a random number of individuals from the population created after the crossover operator in GA, which is now called weed in IWO, to be used in the reproduction operator. These individuals become our initial population for the reproduction operator.

- Reproduction

According to its fitness value, each weed in the population can produce a different quantity of seeds within a specific range of the search space. This means that weeds with lower fitness value produce fewer seeds while those with higher fitness produce more seeds. The calculation of the seeds for each individual is presented in formula 3.5 (Wang et al., 2020)

$$s_i = floor\left(s_{min} + \frac{s_{max} - s_{min}}{f_{max} - f_{min}} \; x \; (f(w_i) - f_{min})\right) \tag{3.5}$$

$f(w_i)$ is the fitness function calculation for weed i, notated as w_i, f_{min} and f_{max} represents the minimum and maximum fitness values obtained so far in the solution, s_{min} and s_{max} are the parameters that define the number of seeds generated by the worst and the best weeds in the population. According to the determined number of seeds, new individuals are obtained by making swaps from the parent.

- Spatial Dispersal

As mentioned in Wang et al. (2020), the resultant seeds are distributed normally around the weeds' fitness value as a mean. The seeds are selected from the feasible solutions around the parent weed w in a neighborhood where the normal distribution has a mean of 0 and a standard deviation std. The standard deviation value calculated as follow in formula 3.6 (Mekni & Chaar Fayech, 2015).

$$\sigma_{iter} = \frac{(iter_{max} - iter)^n}{(iter_{max})^n}\left(\sigma_{initial} - \sigma_{final}\right) + \sigma_{final} \quad (3.6)$$

where max iter denotes the most iterations possible. σ_{iter} represents the current iteration's standard deviation, while n is the nonlinear modulation index. The value of determines the weeds' capacity for exploration.

- Competitive exclusion, and evaluation of fitness of each individual

The number of weeds that always grow from seeds after several generations of reproduction surpass the accessible natural resources. As a result, the offspring and parents are sorted based on a preset fitness value, and those with good fitness move onto the next generation to continue to reproduce. In other words, according to fitness value, all individuals are ranked, the top individuals with the highest fitness value are chosen, and the rest are removed. This process continues until it reaches the maximum population size, which is equal to $PopSize$ in our algorithm.

Algorithm 2: Invasive Weed Optimization

1. Set parameters: population size (PopSize), maximum number of generation (*maxgen*), crossover probability, minimum and maximum number of seeds (s_{min}, s_{max}).
2. Generate an initial population $\boldsymbol{P_0}$ with PopSize individuals. Chromosomes are generated randomly.
3. **Apply** feasibility function.
4. At the beginning, sch_{opt} should be set as empty and its completion time should be assigned as infinite.
5. **For** [t=1,$maxgen$]

5.1 Calculate the population's best and worst fitness values as well as the standard deviation for each iteration.

5.2 **For** [weed w=1,PopSize]

5.2.1 Compute the number of seeds for w depending on its fitness.

5.2.2 **Select** the seeds from the viable solutions around the parent weed **w** in a neighborhood where the normal distribution has a mean of 0 and a standard deviation **std**.

End for

5.3 Evaluate the fitness of function.

5.4 Reduce population to PopSize by rank selection.

5.4 **If** the best individual's completion time from the entire population is shorter than the completion time of sch_{opt}, then sch_{opt} is updated with this individual's record.

End for

6. **Return** the best solution found sch_{opt} so far.

The invasive weed optimization's parameters are initializing at the start, including elements such as population size (PopSize), maximum number of generation (maxgen), minimum and maximum number of seeds (s_{min}, s_{max}), and the initial population $\boldsymbol{P_0}$ is created randomly. The fitness of each individual is determined by the completion time that is calculated using the fitness function. After that the number of seeds for each weed w is calculated depending on their finesses. The seeds are selected from the feasible solutions around the parent weed w in a neighborhood where the normal distribution has a mean of 0 and a standard deviation std that is calculated in the algorithm. As a result, the offspring and parents are sorted based on a preset fitness value, and those with good fitness move onto the next generation to continue to reproduce. The algorithm keeps running until $maxgen$, the number of iterations, is reached.

3.2.6 Genetic Algorithm with Invasive Weed Optimization Reproduction

As we state before, one of the most popular meta-heuristics GA has been used frequently to obtain optimal and nearly optimal solutions to optimization problems in the literature. However, there can be some negative aspects of using the GA. The problem-solving process is trapped in a suboptimal state; many of the operators are no longer able to create offspring who can give better results than their parents. Once the populations are derived for the next generation, the problem previous knowledge is ignored in GA, and the number of offspring produced is not determined by the fitness

values of the parents. Accordingly, studies combine GA with other methods to overcome those negative features of the GA. This is also shown in our literature review for RCPSP. In Valls et al. (2008), they proposed Hybrid Genetic Algorithm (HGA) for resource constrained project scheduling problem (RCPSP). A study by Yannibelli and Amandi (2013) provides insight into multi-objective MS-RCPSP. They combine a multi-objective simulated annealing technique and a genetic algorithm to shorten project durations and increase worker productivity. In Lin, Zhu and Gao (2019), MS-RCPSP is performed with a genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem. They improve their algorithm combining with several simple heuristic's methods.

To solve MS-RCMPSP, a genetic algorithm (GA) was implemented in this thesis by using genetic operators and invasive weed optimization reproduction (IWOR). Our literature review shows that it has many applications in scheduling and project scheduling problems, giving good results in terms of quality and efficiency. Considering the ease of expressing the complex structure of our problem, we decided to use GA in MS-RCMPSP with local and global resources we also used invasive weed optimization reproduction (IWOR) to get a more diverse population and to develop new generations from individuals that give better results. In IWOR algorithm, weeds represent solutions, and all these weeds are involved in the reproduction process. However, the number of weeds to create a new plant is different for each of them. According to the result of the fitness function, which is also the objective function in IWO, the number of plants to be formed from each weed differs, which means the fertilities of each weed is not equal, and more reproduction is achieved from the weeds with better results. These processes continue iteratively until the stopping criterion is met. Unlike GA, IWOR process considers fitness values when creating offspring from parents (Atabaki et al., 2017). The number of offspring to be produced from each parent differs according to their fitness values, which means parents with better fitness values also have higher fertility. Those aspects increase the algorithms' ability to create diversity from higher quality solutions. Wang proposed IWOR to solve multiple travelling salesman problems in 2020; they use IWOR to improve their algorithm quality and efficiency. When they compare their proposed algorithm with other improved GA methods, it is shown that improved GA with IWO reproduction performs better solutions than other. Considering all this information, we decided to take advantage of IWO and decided to combine GA and

IWO. Like Wang et al. (2020), we decided to benefit from IWOR. Both GA and IWO are population-based algorithms, which makes it easy for us to combine two algorithms. In addition, IWO is used in many successful studies today and its use is gradually increasing.

Algorithm 3: Genetic Algorithm with Invasive Weed Optimization Reproduction

1. Set parameters: population size (PopSize), maximum number of generation (*maxgen*), crossover probability (p_c), minimum and maximum number of seeds (s_{min}, s_{max}).
2. Generate an initial population $\boldsymbol{P_0}$ with PopSize individuals. Chromosomes are generated randomly to the selected encoding method.
3. **Apply** feasibility function.
4. At the beginning, sch_{opt} should be set as empty and its completion time should be assigned as infinite.
5. **For** [t=1,$maxgen$]
   - 5.1 Select pairs of individuals (parents/weeds) by using roulette wheel.
   - 5.2 Produce offsprings by applying:
     - 5.2.1 The OX2 crossover operator is applied to each pair of parents with a probability p_c .
     - 5.2.2 **Apply** feasibility function to obtain feasible offsprings.
     - 5.2.3 **Evaluate** the fitness of children
     - 5.2.4 The invasive weed optimization reproduction and feasibility function are used for population $\boldsymbol{R_t}$.
   - 5.3 Evaluate the fitness of function.
   - 5.4 **If** the best individual's completion time from the entire population is shorter than the completion time of sch_{opt}, then sch_{opt} is updated with this individual's record.
   - 5.5 **Select** parents/weeds.
   - 5.6 Add the children to the population as $\boldsymbol{Q_t}$ + $\boldsymbol{R_t}$
   - 5.7 If the copied individual number equals to the determined *duplicated* threshold value.
   - 5.8 Reduce population to PopSize by rank selection.

   End for
6. **Return** the best solution found sch_{opt} so far.

The genetic algorithm's parameters are initializing at the start, including elements such as population size (PopSize), maximum number of generation (maxgen), crossover probability (p_c), minimum and maximum number of seeds (s_{min}, s_{max}), and the initial population is created by using permutation coding randomly. This means that the encoding strategy generates the chromosome of each genotype in the initial population by using invasive weed optimization reproduction. Initializing to fitness calculation, it is necessary to decode all genotypes into phenotypes using the decoding procedure. The fitness of each individual is determined by the completion time that is calculated using the fitness function. The starting populations' $\boldsymbol{P_0}$ fitness value is evaluated using the rank selection method, as explained in the *Selection* section.

Algorithm 2 shows that a new population is generated by using crossover and the reproduction operator. Initially, the individuals are selected with the probability of p_c and population $\boldsymbol{R}_t$ is created. In other words, like Kucuksayacigil and Ulusoy (2020), we determine the population size as $PopSize$, and parent selection and generation of offspring are managed to ensure that a new individual list of size current population ($\boldsymbol{P_t}$)+ $\boldsymbol{Q}_t$ + $\boldsymbol{R}_t$ is obtained. A new population $\boldsymbol{R}_t$ is created with the OX2 crossover operator, and then the fitness values of the population $\boldsymbol{R}_t$ are calculated. In the next step the new offspring are produced through IWOR, a new population $\boldsymbol{Q}_t$ is created by using population $\boldsymbol{R}_t$. A new population consists of a recently formed population $\boldsymbol{R}_t$ and population $\boldsymbol{Q}_t$. In this algorithm like Atabaki et al., (2017), to increase diversity, copied solutions in the temporary population ($\boldsymbol{P_t} + \boldsymbol{R}_t$ + $\boldsymbol{Q}_t$) are replaced with the swap operator. Here, we use *duplicated* threshold value; if there are two duplicate solutions in small populations, a swap operator is used, and in medium and large populations, this number is four. In other words, to prevent premature convergence brought on by a dramatic rise in the population and due to a rapid increase in the number of copies of the fittest individuals, we employed this method. According to this method, duplicates solutions will change if the number of copies of individuals exceeds a preset upper bound and the produced seeds are replaced by swap operator.

The preeminent individuals in the new population ($\boldsymbol{R}_t$ + $\boldsymbol{Q}_t$) are selected for the current population $\boldsymbol{P}_t$ as the number of $PopSize$. The algorithm keeps running until $maxgen$, the number of iterations, is reached.

The major difference between the proposed genetic algorithm and genetic algorithm with invasive weed optimization reproduction is the way to generating offspring. Regarding crossover and mutation, standard genetic algorithms use both operators to produce offspring, while only the crossover operator is used in the genetic algorithm with invasive weed optimization reproduction. The selected individuals in each population are mutated by mutation with a probability of p_m in the proposed GA but the reproduction mechanism employed in GA-IWOR has effectively neutralized the influence of the mutation operator. The results achieved through the GA-IWOR is better than those obtained via conventional GA methods and IWO, and these outcomes are comprehensively discussed and presented in Chapter 4.

Chapter 4:
EXPERIMENTAL RESULTS

This section presents the instance attributes and the computational results of the genetic algorithm (GA) with invasive weed optimization reproduction (IWOR). Firstly, we present the instance attributes that we generate to evaluate the algorithms performance and analyze the key parameters of the problem to successfully find a solution. The second part includes the performance of the proposed algorithms GA with IWOR and standard genetic algorithm with the 30 test instances.

4.1 Instance Generation

In this section, we describe how the data is generated to test our algorithm's performance. In the literature, no data fully address MS-RCMPSP with local and global resources. Hence, we had to create a data set to use in our study. An instance generator called iMOPSE dataset instance generator, which was developed by the Wroclaw University of Technology, helped us to create a single project data. The instance generator enables us to create new instances of MS-RCPSP. It also allows us to determine the key elements: tasks, resources, precedence relations, and skills determination of any project. The important aspect for tasks is its duration. The data generator enables to set a value between the minimum and maximum duration. The only condition is the range must be positive values. The number of precedence relations; we consider the number of preceding tasks for each task in our projects to identify precedence relations and ensure the proper order of completion. The resources have two input parameters: salary and number of resources. We can define how many resources would be used in a project, but the standard and overtime salary rates are not considered because the only objective considered in this study is minimizing the makespan. In addition, the resources have skill attributes to define them; the data generator provides us with two parameters: the number of skill types in a project and the number of skills owned by the resource. The skills generated within the project are assessed independently from each other. Only one skill is required for the task in the generator,

but the generator determines how many skills and skill levels for a resource at selected intervals. While combining the single project data we obtained through the iMOPSE dataset instance generator, we benefited from the idea of the generation of MISTA Challenge (2013) data. The data obtained in this challenge is multi-mode multi-project data, but the way they generate it is like the main idea the data generation for MISTA Challenge (2013). The data used in this competition were also obtained by combining single project data. They create a global file and a file of every single project. The global file provides the path of every single project file. It also has information on global resources' skill and skill levels and the total number of projects included in that instance. The global data file scheme can be seen in Figure 4.1.

Total Project Number
Path of Project 1
Path of Project 2
Path of Project 3
Path of Project 4
…
Path of Project H
Global Resources Skill and Skill Levels

Figure 4.1: Global data file scheme

Each project is generated in separate files (e.g. file1.txt), containing the number of tasks, tasks durations, local resources skill with skill levels and precedence relationships among the tasks. All those files are obtained through the iMOPSE dataset instance generator. Example of global and project files can be found in Appendices A and B, respectively.

30 instances are randomly generated to evaluate the developed algorithms' performance. These instances have four attributes based on the number of tasks $N=\sum_{h\in\eta}|J_h|$, skills (S), skill levels (U) and workforces (R) and divided into three categories: small, medium, and large. We define multi-projects, including 50 to 200 tasks as small, 200 to 350 as medium, and 350 to 500 as large instances. Table 4.1 describes these instances attributes and sizes.

Table 4.1: The generated instances features

Instances	N	S	U	R
1	80	3	3	10 local and 1 global resources
2	115	3	3	20 local and 1 global resources
3	110	3	3	20 local and 1 global resources
4	100	4	4	10 local and 1 global resources
5	120	4	4	20 local and 1 global resources
6	80	4	4	6 local and 1 global resources
7	90	3	3	8 local and 1 global resources
8	110	3	3	20 local and 1 global resources
9	100	3	3	20 local and 1 global resources
10	115	4	4	20 local and 1 global resources
11	250	4	4	40 local and 2 global resources
12	300	3	3	40 local and 3 global resources
13	300	4	4	45 local and 3 global resources
14	220	3	3	30 local and 3 global resources
15	215	3	3	30 local and 2 global resources
16	210	3	3	30 local and 2 global resources
17	200	4	4	30 local and 2 global resources
18	300	3	3	40 local and 3 global resources
19	300	3	3	30 local and 3 global resources
20	200	4	4	20 local and 2 global resources
21	500	3	3	45 local and 5 global resources
22	440	4	4	50 local and 5 global resources
23	500	4	4	50 local and 5 global resources
24	400	4	4	45 local and 5 global resources
25	500	3	3	50 local and 5 global resources
26	500	4	4	35 local and 5 global resources
27	400	3	3	35 local and 5 global resources
28	440	4	4	45 local and 5 global resources
29	500	4	4	50 local and 5 global resources
30	400	3	3	50 local and 5 global resources

The duration of each task was uniformly distributed in the interval of [8, 40], and the total number of tasks generated was from 60 to 500. The number of skills (S) and skill levels (U) are changed randomly as 3 and 4. The number of local renewable resources is generated according to load measure, and the number of global resources is

assigned considering the size of the projects: for small instances one global resource, medium instances two and three and large instances five global resources are assigned.

4.2 Tuning the Algorithm Parameters

The proposed method includes a genetic algorithm (GA) and invasive weed optimization reproduction (IWOR) and there is no effortless way to configure parameters. Since each algorithm contains stochastic rules, we must consider many parameters, such as population size PopSize, the maximum number of iterations $maxgen$, the maximum and the minimum number of seeds S_{min} and S_{max} and crossover probability p_c. We select appropriate parameters for the algorithm by examining the instances separately as small, medium, and large. Ten problem instances are included in experimental instances selected for parameter tuning. To determine the appropriate parameters, we design a factorial analysis on a small pilot set of 10 problem instances. There are several parameters in the proposed algorithm, so the analysis is limited by the values used in similar studies in the literature. The parameter configurations used in similar studies in the literature can be seen in Table 4.3.

Goçalves, Mendes and Resende (2008) demonstrate that for the population size, they obtained good results by indexing it to the size of the problem. They use a small population size for small instances and a larger population size for large instances. They propose to use a minimum of $0.2\ x\ N$ or 250 as a population size. Like Goçalves, Mendes and Resende (2008), we determined the population size as $0.2\ x\ N$ to set best parameter configuration. Crossover probability is determined as values in the literature's range of [0.7, 0.9]. In Table 4.3, it is seen that this value range is always chosen in studies.

Table 4.2: Tian and Yuan (2018) test parameters

Parameters	**Range of Values**
$\boldsymbol{maxgen}$	[5,200]
$\boldsymbol{P_c}$	[0.02, 1]
PopSize	[20,120]

Tian and Yuan (2018) suggest parameter configurations for RCPSP using the design of experiments (DOE). They focused on the resource-constrained project

scheduling problem parameters and examined the genetic algorithm's population size PopSize, mutation probability p_m, and crossover probability p_c. They used an existing RCPSP library called PSPLIB to apply the experiment. They divided instances into four categories, which have 30, 60, 90 and 120 tasks in total. The ranges they tested for these datasets are given in Tables 4.2 and 4.3.

Table 4.3: Parameter values used in the literature

	Parameters	**Values**
Solving resource-constrained project scheduling problem via genetic algorithm (liu, j., liu, y., shi, y., and li, j. (2020))	$\boldsymbol{PopSize}$	30 and 120 for 60 and 240 tasks
	P_c	0.70 and 0.80
	P_m	0.10,0.15,0.20
	$\boldsymbol{maxgen}$	1,5,50 K schedules are generated
A hybrid genetic algorithm for the resource-constrained project scheduling problem (valls, v., ballestín, f., and quintanilla, s. (2008))	$\boldsymbol{PopSize}$	100 and 120 for 50,100 and 400 tasks
	P_c	0.7,0.8,0.9
	P_m	0.05
	$\boldsymbol{maxgen}$	100,125,150,250
A genetic algorithm for the resource constrained multi-project scheduling problem (gonçalves, j.f.,mendes, j.,and resende,m. (2008))	$\boldsymbol{PopSize}$	min(0.2xNumber of activities in the multi-project,250)
	P_c	0.70,0.75,0.80
	P_m	0.2
	$\boldsymbol{maxgen}$	50
A genetic programming hyper-heuristic approach for multi-skill resource constrained project scheduling problem (lin, zhu and gao, 2019)	$\boldsymbol{PopSize}$	60 and 100 for 100 and 200 tasks
	P_c	Not used
	P_m	0.15;100, 0.1;200
	$\boldsymbol{maxgen}$	800
Efficient genetic algorithm for resource-constrained project scheduling problem (wang et al., 2010)	$\boldsymbol{PopSize}$	80
	P_c	0.7
	P_m	0.05
	$\boldsymbol{maxgen}$	5,24,244

In addition, the IWOR mechanism has its own parameters. Since most of the parameters like PopSize and maxgen are shared with the genetic algorithm, the same parameter ranges are tested for IWOR. Then, the parameters we had to decide on are s_{min} and s_{max} , where s_{min} and s_{max} are the parameters that define the number seeds generated by the worst and the best weeds in the population. For determining s_{min} and s_{max} values, a literature review is performed Atabaki et al., (2017) and Maghsoudlou and Sadeghzade (2016) both suggest using s_{min} = 5 and s_{max}= 10. We also tried s_{min} = 1 and s_{max}= 5 because the CPU time was increasing considerably between the two alternatives that are highly recommended in the literature.

Considering the literature review, we restricted the parameter search space using the commonly used ranges. Table 4.4 contains the ranges of the value selected for parameter tuning in this study. Small, medium, and large instances were selected for 10 pilot tests, and all instances run ten times. Makespans are calculated by averaging all these ten runs for each instance and we obtain total 100 runs in total. The best parameter configurations were determined separately for small, medium, and large problems to be more accurate.

Parameters	**Values**		
	Small	**Medium**	**Large**
$\boldsymbol{maxgen}$	50	100	120
P_c	0,9	0,8	0,9
P_m	0.05	0.05	0.05
$PopSize$	20	80	120
S_{min}	5	5	5
S_{max}	10	10	10

Table 4.4: The suggested parameters for small, medium, and large instances

According to the results in Appendices C to K, the trend of each parameter level for large, medium, and small instances can be depicted from Figure 7.6 to Figure 7.13. It can be found that in the instances with medium and large sizes, the number of maximum generations and s_{min} and s_{max} are the most significant among the four parameters, followed by the population size PopSize, while the crossover probability p_c is the least effective. Specifically, it can be observed Appendices C to K that a larger value of

generation results in smaller makespan values, but the downtrend becomes not so significant when $maxgen$ is set to 50, 100 and 200 for small, medium, and large instances, respectively. As for the population size, $PopSize$ is almost reaching the best value when it is set to 20 or 22 for small, 60 for medium and 100 for large instances. The selected parameter values can be seen in Table 4.4.

4.3 Experimental design

We achieved impressive results through data runs, providing valuable insights on algorithms' performance in terms of CPU time and makespan. In the following sections, the performances of the algorithms are given in Table 4.5, 4.6, 4.7 and 4.8. Since there is no benchmark dataset that address our problem, that is MS–RCMPSP with global and resources, numerical experiments are carried out on the new benchmark dataset generated.

The new dataset consists of 30 multi-project instances with the size ranging from 80 tasks to 500 tasks. The GA with IWOR algorithm is implemented in MATLAB R2023a and operated on a 1,6 GHz Dual-Core Intel Core i5 with 16 GB RAM. For each instance, the algorithm runs 30 times independently. In small data sets where we obtain 50 generations in each run, we obtain a total of 1500 generations with 30 runs; in medium data sets where we obtain 100 generations, we obtain a total of 3000; and in large data sets where we obtain 120 generations, we obtain a total of 3600 generations. The best values, averages, and standard deviations of the makespan and CPU values of the 30 runs we take for each sample are given in the tables where we show the results. The results presented in Table 4.5, 4.6, 4.7 and 4.8 demonstrate the performance of the GA-IWOR algorithm by comparing with GA and IWO algorithms. The IWO algorithm consistently achieved the smallest standard deviation value 0.874 and this indicates a high level of precision and consistency in the algorithm's results. However, there is no significant difference between it and other algorithms. GA-IWOR comes after IWO with a value of 0.89, and GA place with 0.905 last. Furthermore, the generated dataset had an average makespan of 296.57 hours, with an average standard deviation of 0.89 by using GA-IWOR algorithm. These results suggest that the GA-IWOR algorithm can produce best solutions with consistently low variance in its results. Experimental tests have confirmed the successful performance of the GA-IWOR algorithm when compared

to standard GA and IWO algorithms regarding makespan optimization. The GA-IWOR algorithm is a promising new approach for optimization and decision-making. Current studies indicate it is beneficial across several industries, but further research is required to understand its potential and constraints.

Table 4.5: Computational results of makespan for GA

	Algorithm 1: GA		
Instance	Best	Average	Std
1	240	240.73	0.340
2	258	260.70	0.499
3	163	163.87	2.098
4	342	343.87	0.964
5	298	299.00	0.929
6	409	409.73	0.300
7	330	332.73	1.024
8	248	248.90	2.089
9	169	169.53	1.949
10	261	265.63	0.943
11	245	246.37	0.12
12	265	265.67	1.454
13	163	165.67	0.605
14	566	569.47	0.806
15	490	491.97	0.23
16	192	193.13	0.748
17	394	394.93	1.231
18	196	196.20	0.427
19	301	301.87	1.093
20	201	201.13	1.136
21	157	158.07	0.373
22	185	185.90	0.540
23	381	381.83	1.140
24	288	289.10	1.258
25	224	225.03	0.100
26	432	433.13	0.249
27	384	384.10	0.907
28	603	603.93	0,690
29	402	403.67	1,090
30	203	205.73	0.340
Average	299,67	301.11	0.905

Table 4.6: Computational results of makespan for IWO

Algorithm 2: IWO			
IWO	Best	Average	Std
1	238	239.33	0.650
2	254	254.3	0.900
3	161	162.37	0.547
4	342	343.27	0.964
5	298	299.37	1.602
6	408	409.27	1.436
7	330	332.1	1.446
8	247	247.47	0.499
9	166	166.27	0.442
10	258	260.63	2.089
11	242	241.2	1.833
12	265	265.2	0.600
13	160	162.37	1.110
14	566	569.93	1.209
15	490	493.53	0.846
16	192	192.9	0.300
17	392	392.93	1.569
18	193	193.3	0.900
19	298	298.37	0.948
20	189	193.3	2.312
21	157	157.2	0.600
22	184	184.9	0.300
23	381	381.9	0.300
24	287	288.2	0.600
25	223	223.9	0.300
26	431	431	0.000
27	383	383.67	0.471
28	602	602.93	0.249
29	396	397.2	0.970
30	202	203.13	0.991
Average	297,83	299.048	0.874

In summary, its higher performance in makespan optimization compared to standard GA and IWO algorithms are validated by experimental experiments. The GA-IWOR algorithm has potential for optimization and decision-making across a range of industries, but further study is required to fully understand its strengths and weaknesses. Therefore, in this section, more than one performance measure is used to understand the performance of the algorithms and to show whether the results are statistically significant.

Table 4.7: Computational results of makespan for GA- IWOR

Algorithm 3: GA-IWOR			
Instances	Best	Average	Std
1	236	237.53	0.846
2	252	255.37	1.719
3	160	160.80	0.600
4	339	340.80	1.470
5	298	298.50	1.118
6	405	407.47	1.910
7	328	329.53	1.310
8	245	246.03	1.248
9	163	164.90	1.446
10	257	257.47	0.623
11	239	240.13	1.024
12	263	265.00	0.730
13	159	163.87	0.818
14	566	566.47	1.431
15	489	489.17	0.582
16	190	191.90	0.700
17	392	392.13	0.340
18	191	191.57	0.616
19	297	298.23	1.055
20	197	198.40	1.332
21	156	156.07	0.249
22	183	184.20	0.653
23	378	379.43	1.564
24	283	283.70	1.187
25	221	222.83	0.637
26	431	431.89	0.236
27	382	382.00	0.000
28	600	601.63	0.948
29	396	397.10	1.287
30	201	201.83	0.921
Average	296.57	297.87	0.890

The CPU times consumed by the GA-IWOR are also presented in Table 4.8. However, it is worth noting that in some instances, where the number of tasks is as high as 500, the algorithm seems to require a slightly longer processing time when comparing with the GA results in Table 4.8. Despite this, it is clear overall; the standard algorithm is mostly quite effective and efficient in terms of the CPU time required for its execution, when comparing the GA-IWOR algorithm

Table 4.8: Computational results of CPU for GA, IWO and GA-IWOR

Average (CPU time s)			
Instances	GA	IWO	GA-IWOR
1	124.375	126.789	129.001
2	124.589	138.985	130.018
3	125.106	140.124	129.018
4	126.789	150.673	137.543
5	125.106	144.672	129.021
6	124.598	128.356	132.256
7	124.680	136.416	132.318
8	125.346	133.255	129.250
9	125.128	125.456	129.456
10	127.192	142.146	136.089
11	125.987	125.789	129.892
12	298.789	303.780	307.440
13	438.567	440.239	445.886
14	137.748	148.790	159.021
15	293.905	305.167	301.239
16	294.150	331.896	351.676
17	292.194	393.127	309.012
18	285.789	289.789	301.786
19	345.890	363.876	353.412
20	288.145	385.393	301.216
21	825.178	845.680	846.124
22	945.021	965.789	963.189
23	914.987	925.756	931.172
24	282.062	301.456	288.640
25	860.378	860.459	867.986
26	765.441	784.239	789.654
27	261.670	270.125	273.167
28	373.934	393.257	397.572
29	945.331	995.256	1.001,176
30	389.921	389.553	401.367
Average	**348.267**	**369.876**	**364.820**

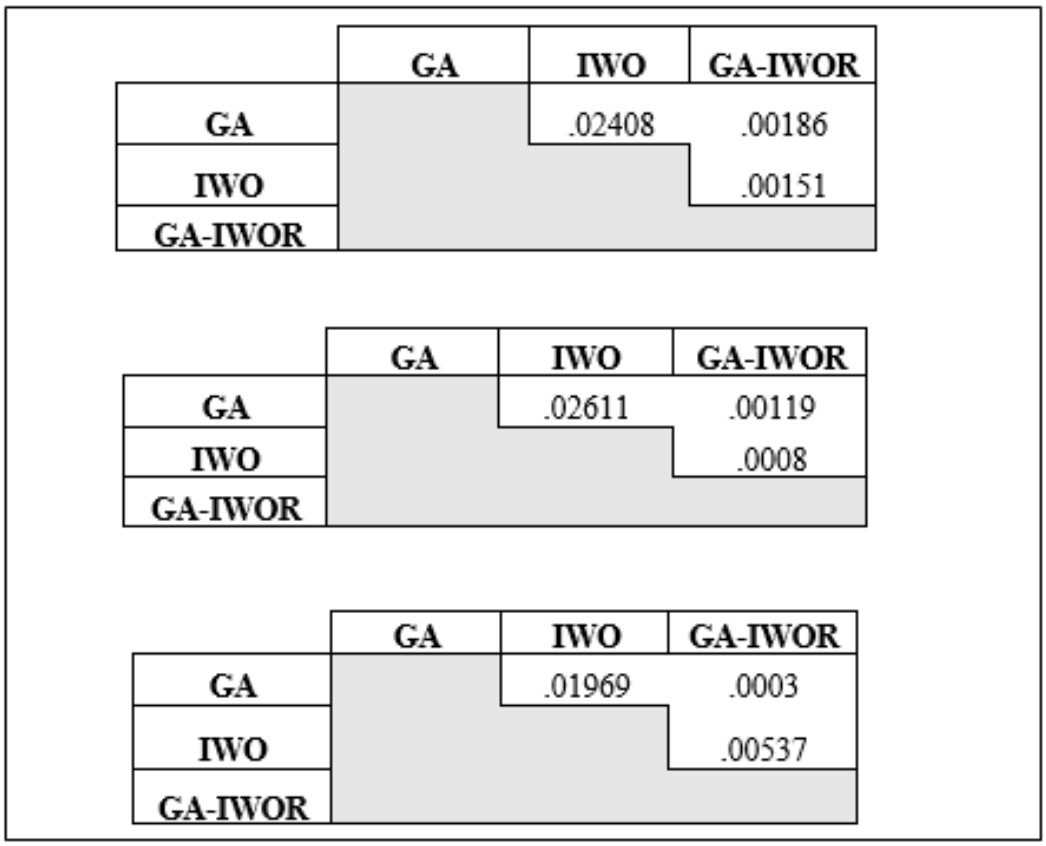
In some cases, algorithm performance is measured by how quickly it can process a given amount of data. When we compare it with IWO, GA-IWOR performs better than IWO in terms of CPU time. In general, in many cases, runs were completed in less time than IWO or in approximately equal time. In this case, a decision maker who wants to obtain the shortest makespan should choose GA-IWOR. The most successful algorithm in CPU time was GA, which is an expected result because a hybrid algorithm can be expected to run longer than the standard algorithm. However, the difference is moderate, even in large samples. IWO was successful in terms of standard deviation value and consistency. IWO should be selected if the decision maker wants a more consistent result. However, GA-IWOR will be one of the best choices, which does not give bad results in all three performance indicators and even gives the shortest project completion time. In the Table below, the relaxed values consider a case where all resources can support all projects as global resources; all projects are a shared resource pool. Again, 30 runs for each instance were taken with GA-IWOR and average values are presented. The result to be seen here is that the total project durations have been shortened considerably, which shows that it is much more challenging to find the shortest project completion times in project environments with local and global resources.

Table 4.9: Computational result of total makespan with a shared resource pool

Total makespan with a shared resource pool									
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
151.23	120	93.45	268.78	249.12	409.02	210	240.36	136.30	169.67
11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
212.02	197.67	141.34	353.23	430.34	191.78	207	151.89	256.97	159.35
21	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
132.19	153	348.73	279.03	220.57	425.29	370.20	598.19	385.56	189.38

This is an expected result for relaxing. For example, consider a team with five local and two global resources per project; five projects will be scheduled simultaneously. In our problem, while the resource we are looking for is 7 for each project, we have 27 resources to find a talented resource in a shared resource pool.

Figure 4.2: p-value for the paired two dependent means for small, medium and large instances



	GA	IWO	GA-IWOR
GA		.02408	.00186
IWO			.00151
GA-IWOR			

	GA	IWO	GA-IWOR
GA		.02611	.00119
IWO			.0008
GA-IWOR			

	GA	IWO	GA-IWOR
GA		.01969	.0003
IWO			.00537
GA-IWOR			

We create a t-test to test whether the results obtained from three different algorithms create differences worthy of statistical interpretation. The results of the algorithms were subjected to two dependent means t-tests in pairs, and the p-values of this test are presented in Figure 4.2. It can be seen from the p-values obtained are less than 0.05, the significance level (p) value we chose in the test, and the results are statistically significant. (The result is significant at $p < .05$.) The statistical significance of the performance differences is indicated by our t-test results in our table for GA compared to IWO, GA compared to GA-IWOR, and IWO compared to GA-IWOR. According to the results of this test, we can say that IWO performs better than GA and GA-IWOR performs better than IWO in terms of makespan.

Chapter 5:
CONCLUSION AND FUTURE WORKS

This thesis presents a newly developed project management problem that has not been explored previously in the literature. The project environment of a global automotive company has been examined and this problem has been created. Specifically, we tackle the challenge of efficiently scheduling resources that possess different skill levels and are placed in different geographic locations. To address this problem, we propose a multi-skill resource-constrained multi-project scheduling problem that considers both global and local resources. The goal of our approach is to assign resources and create project schedules in a way that minimizes the overall completion time of the project. In addition to introducing this new problem, our study also highlights the importance of incorporating real-life factors into project management decision-making. We emphasize the need to consider information on task durations, the skills and skill levels of resources, and the fact that multiple projects are often executed simultaneously. Moreover, we recognize that resources may be in different regions, which further complicates the scheduling process.

This thesis has several reasons for extending the standard RCPSP to MS-RCMPSP. Those reasons include the high wages of employees in the markets where the company is located and the need for a team to support companies globally. This thesis presents two decisions to assist those decision-making processes for the companies. The companies can assign resources to the right tasks according to their skills and skill levels and schedule them within the project. While making these decisions, we aim to complete all projects as soon as possible. Although resources are one of many constraints in this problem, tasks within projects also have constraints. While scheduling the tasks, attention was also paid to their precedence relationships.

Similarly, in Heimerl and Kolisch (2009), they formulated scheduling and staffing for multiple projects with a multi-skilled workforce, but they assumed that the work packages were given; that is, the task schedules within the projects were given, and only the beginning and the ending of the projects were not known. Simultaneously, resources

are used as internal and external in the problem. In this paper, task scheduling within the projects is also considered. Especially in recent years, the multi-project scheduling problem has been studied widely, but whether the resources are local or global has yet to be considered. Haroun et al. (2022) published the last work on MS-RCMPSP; they present the problem with interruption allowed while executing tasks, and there is no distinction such as global or local resources. As seen in our literature review, there has yet to be a similar study.

We proposed a genetic algorithm with invasive weed optimization reproduction. The algorithm enables us to find the optimal or near-optimal solution to real-world problems that are NP-hard, which means the optimal solution cannot be found in polynomial time. Since there is no benchmark data in the literature, we generated our problem instances to evaluate algorithm performance. As stated by Cai, Chang and Zhang (2020), there are few studies on IWO's use in RCPSP, even though it has been successfully employed in numerous areas. Maghsoudlou and Sadeghzade (2016) use an improved invasive weed algorithm to introduce a multi-skill multi-mode resource-constrained project scheduling problem. Then, Cai, Chang and Zhang (2020) proposed using improved invasive weed optimization for the standard RCPSP. We show how to implement invasive weed optimization reproduction with GA in multi-skill resource-constrained multi-project scheduling problems. Our results showed that IWOR eliminates the effectiveness of the mutation operator in the genetic algorithm. The results showed that when there is no mutation operator, better results are obtained with invasive weed optimization reproduction.

To improve our work and make our model more realistic, we could consider incorporating processing times for different human resources' capabilities into the MS-RCMPSP model. This would allow us to simulate real-world project scheduling systems more accurately and make better decisions. Finally, we can explore the use of invasive weed optimization in other project scheduling problems that we have examined in our literature review. This technique has shown great promise in a variety of different contexts and could potentially be applied to a range of RCPSP problems with excellent results. By wishing for these directions, we can continue explaining our processes and develop better and more effective solutions for project scheduling in the future. Future research should consider how a local resource's competence level may vary when doing a global task. In other words, local resources may act as a global resource in some cases,

and how skills and skill levels may vary in that situation can be considered. In addition, the required skill for a task can be provided additively. In other words, two resources with 2 and 3 skill levels can be assigned, for example, for a task that needs five skill levels from a skill.

In this study, the projects considered that start at the same time, such as the roll-out of a digital product to different countries, the extension of a newly developed application to pilot markets or the setting of developments to different branches within a company. However, projects with different start dates can also be considered; similar examples are available in the literature. In this study, we can solve this problem by adding the starting time attribute to the tasks or by rerunning without making any changes to the algorithm.

Chapter 6:

BIBLIOGRAPHY

Anantkumar, J., Umbarkar, S., & Sheth, M. (2015). Crossover operators in Genetic Algorithms: A Review. ICTACT Journal on Soft Computing, 06(01), 1083–1092. https://doi.org/10.21917/ijsc.2015.0150

Arabnejad, V., Bubendorfer, K., & Ng, B. (2017). Scheduling deadline constrained scientific workflows on dynamically provisioned cloud resources. Future Generation Computer Systems, 75, 348–364. https://doi.org/10.1016/j.future.2017.01.002

Atabaki, M. S., Mohammadi, M., & Naderi, B. (2017). Hybrid genetic algorithm and invasive weed optimization via priority-based encoding for location-allocation decisions in a three-stage supply chain. Asia-Pacific Journal of Operational Research, 34(02), 1750008. https://doi.org/10.1142/s0217595917500087

Bellenguez-Morineau, O. (2007). Methods to solve multi-skill project scheduling problems. 4OR, 6(1), 85–88. https://doi.org/10.1007/s10288-007-0038-4

Besikci, U., Bilge, U., & Ulusoy, G. (2012). Resource dedication problem in a multi-project environment. Flexible Services and Manufacturing, 25. https://link.springer.com/article/10.1007/s10696-012-9140-9

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

De Boer, Ronald. (1998). Resource-constrained Multi-Project Management - A Hierarchical Decision Support System.

Cai, W., Chen, H., & Zhang, J. (2020). An enhanced invasive weed optimization in resource-constrained project scheduling problem. 2020 11th International Conference On Awareness Science and Technology (ICAST). https://doi.org/10.1109/icast51195.2020.9319493

Caramia, M. (2020). Project management and scheduling. Annals of Operations Research, 285, 1–8. https://doi.org/10.1007/s10479-019-03414-9

Chen, R., Liang, C., Gu, D., & Leung, J. Y.-T. (2017). A multi-objective model for multi-project scheduling and multi-skilled staff assignment for IT product

development considering competency evolution. International Journal of Production Research, 55(21), 6207–6234. https://doi.org/10.1080/00207543.2017.1326641

Chieng, H. H., & Wahid, N. (2014). A performance comparison of genetic algorithm's mutation operators in N-cities open loop traveling salesman problem. Advances in Intelligent Systems and Computing, 89–97. https://doi.org/10.1007/978-3-319-07692-8_9

Čorić, R., Đumić, M., & Jakobović, D. (2017). Complexity comparison of integer programming and genetic algorithms for resource-constrained scheduling problems. 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, pp. 1182-1188. doi: 10.23919/MIPRO.2017.7973603

Cui, L., Liu, X., Lu, S., & Jia, Z. (2021). A variable neighborhood search approach for the resource-constrained multi-project collaborative scheduling problem. Applied Soft Computing, 107, 107480. https://doi.org/10.1016/j.asoc.2021.107480

Dai, H., Cheng, W., & Guo, P. (2018). An Improved Tabu Search for Multi-skill Resource-Constrained Project Scheduling Problems Under Step-Deterioration. Arabian Journal for Science and Engineering, 43, 3279–3290. https://doi.org/10.1007/s13369-017-3047-4

Dang Quoc, H., Nguyen The, L., & Nguyen Doan, C. (2022). An effective hybrid algorithm based on particle swarm optimization with migration method for solving the Multiskill resource-constrained project scheduling problem. Applied Computational Intelligence and Soft Computing, 2022, 1–12. https://doi.org/10.1155/2022/6230145

Goncharov, E. N., & Leonov, V. V. (2017). Genetic algorithm for the resource-constrained project scheduling problem. Automation and Remote Control, 78(6), 1101–1114. https://doi.org/10.1134/s0005117917060108

Gonçalves, J. F., Mendes, J. J. M., & Resende, M. G. C. (2008). A genetic algorithm for the resource-constrained multi-project scheduling problem. European Journal of Operational Research, 189(3), 1171–1190. https://doi.org/10.1016/j.ejor.2006.06.074

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

Haroune, M., Dhib, C., Neron, E., et al. (2023). Multi-project scheduling problem under shared multi-skill resource constraints. TOP, 31, 194–235. https://doi.org/10.1007/s11750-022-00633-5

Hartmann, S. (2002). A self-adapting genetic algorithm for project scheduling under resource constraints. Naval Research Logistics, 49(5), 433–448. https://doi.org/10.1002/nav.10029

Heimerl, C., & Kolisch, R. (2009). Scheduling and staffing multiple projects with a multi-skilled workforce. OR Spectrum, 32(2), 343–368. https://doi.org/10.1007/s00291-009-0169-4

Holland, J. H. (1973). Genetic algorithms and the optimal allocation of trials. SIAM Journal on Computing, 2(2), 88–105. https://doi.org/10.1137/0202009

Hosseinian, A. H., & Baradaran, V. (2021). A two-phase approach for solving the multi-skill resource-constrained multi-project scheduling problem: A case study in the construction industry. Engineering, Construction and Architectural Management. https://doi.org/10.1108/ecam-07-2019-0384

Kadri, R. L., & Boctor, F. F. (2018). An efficient genetic algorithm to solve the resource-constrained project scheduling problem with transfer times: The Single Mode Case. European Journal of Operational Research, 265(2), 454–462. https://doi.org.10.1016/j.ejor.2017.07.027

Kadrou, Y., & Najid, N. M. (2006). Tabu search algorithm for the MRCPSP with multi-skilled personnel, Computational Engineering in System Application, IMACS Multi-Conference, 2(4-6), 1302-1309.

Kolisch, R., Sprecher, A., & Drexl, A. (1995). Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science, 41(10), 1693–1703.

Kora, P., & Yadlapalli, P. (2017). Crossover operators in Genetic Algorithms: A Review. International Journal of Computer Applications, 162(10), 34–36. https://doi.org/10.5120/ijca2017913370

Kucuksayacigil, F., & Ulusoy, G. (2020). Hybrid genetic algorithm for bi-objective resource-constrained project scheduling. Frontiers of Engineering Management, 7(3), 426–446. https://doi.org/10.1007/s42524-020-0100-x

Li, Y.-Y., Lin, J., & Wang, Z.-J. (2021). Multi-skill Resource Constrained Project scheduling using a multi-objective discrete Jaya algorithm. Applied Intelligence, 52(5), 5718–5738. https://doi.org/10.1007/s10489-021-02608-8

Lim, S. M., Sultan, A. B., Sulaiman, M. N., Mustapha, A., & Leong, K. Y. (2017). Crossover and mutation operators of genetic algorithms. International Journal of Machine Learning and Computing, 7(1), 9–12. https://doi.org/10.18178/ijmlc.2017.7.1.611

Lin, J., Zhu, L., & Gao, K. (2020). A genetic programming hyper-heuristic approach for the multi-skill resource-constrained project scheduling problem. Expert Systems with Applications, 140, 112915. https://doi.org/10.1016/j.eswa.2019.112915

Maghsoudlou, H., Afshar-Nadjafi, B., & Niaki, S. T. (2016). A multi-objective invasive weeds optimization algorithm for solving multi-skill multi-mode Resource Constrained Project Scheduling problem. Computers and Chemical Engineering, 88, 157–169. https://doi.org.10.1016/j.compchemeng.2016.02.018

Mehrabian, A. R., & Lucas, C. (2006). A novel numerical optimization algorithm inspired from weed colonization. Ecological Informatics, 1(4), 355–366. https://doi.org.10.1016/j.ecoinf.2006.07.003

Mekni, S., & Chaar Fayech, B. (2015). Multi-objective flexible job shop scheduling using a modified invasive weed optimization. International Journal on Soft Computing, 6(1), 25–36. https://doi.org.10.5121/ijsc.2015.6103

Myszkowski, P. B., & Laszczyk, M. (2022). Investigation of benchmark dataset for many-objective multi-skill resource constrained project scheduling problem. Applied Soft Computing, 127, 109253. https://doi.org/10.1016/j.asoc.2022.109253

Ozdamar, L. (1999). A genetic algorithm approach to a General Category Project Scheduling Problem. IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews), 29(1), 44–59. https://doi.org.10.1109/5326.740669

Pan, Q.-K., Fatih Tasgetiren, M., & Liang, Y.-C. (2008). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. Computers and Operations Research, 35(9), 2807–2839. https://doi.org.10.1016/j.cor.2006.12.030

Sharma, P., Wadhwa, A., & Komal, K. (2014). Analysis of selection schemes for solving an optimization problem in genetic algorithms. International Journal of Computer Applications, 93(11), 1–3. https://doi.org.10.5120/16256-5714

Suresh, M., Dutta, P., & Jain, K. (2015). Resource-constrained multi-project scheduling problem with Resource Transfer Times. Asia-Pacific Journal of Operational Research, 32(06), 1550048. https://doi.org.10.1142/s0217595915500487

Ulusoy, G., & Ulusoy, G. (2002). Proje planlamada kaynak kısıtlı çizelgeleme.
Valls, V., Ballestín, F., & Quintanilla, S. (2008). A hybrid genetic algorithm for the resource-constrained project scheduling problem. European Journal of Operational Research, 185(2), 495–508. https://doi.org.10.1016/j.ejor.2006.12.033

Valls, V., Perez, A., & Quintanilla, S. (2009). Skilled workforce scheduling in service centers. European Journal of Operational Research, 193, 791-804.
Viana, M. S., Junior, O. M., & Contreras, R. C. (2020). An improved local search genetic algorithm with multi-crossover for job shop scheduling problem. Artificial Intelligence and Soft Computing, 464–479. https://doi.org.10.1007/978-3-030-61401-0_43

Walter, M., & Zimmermann, J. (2016). Minimizing average project team size given multi-skilled workers with heterogeneous skill levels. Computers and Operations Research, 70, 163–179. https://doi.org.10.1016/j.cor.201

Wang, H., Li, T., and Lin, D. (2010). Efficient genetic algorithm for resource-constrained project scheduling problem. Transactions of Tianjin University, 16(5), 376–382. https://doi.org/10.1007/s12209-010-1495-y

Wang, J., Zhang, J., and Wang, X. (2018). A data driven cycle time prediction with feature selection in a semiconductor wafer fabrication system. IEEE Transactions on Semiconductor Manufacturing, 31(1), 173

Wang, Z., Fang, X., Li, H., and Jin, H. (2020). An improved Partheno-genetic algorithm with reproduction mechanism for the multiple traveling salesperson problem. IEEE Access, 8, 102607–102615. https://doi.org/10.1109/access.2020.2998539

Chapter 7:

APPENDIX

7.1 Appendix A. The sample of global data file

```
10
file72_A.txt
file73_A.txt
file74_A.txt
file75_A.txt
file76_A.txt
file77_A.txt
file78_A.txt
file79_A.txt
file80_A.txt
file81_A.txt
2 2 3 3 0 3 1 3
1 3 0 2
1 2 3 2 0 1
3 3
2 1 3 3 1 2
```

Figure 7.1.1: The sample of global data file

7.2 Appendix B. The sample of project data file

```
============================================================
File name: /Users/user/Desktop/Data_Set_C\file72_A.def
Creation date: Sun Nov 27 22:21:09 EET 2022
Website: http://imopse.ii.pwr.edu.pl/
Reference:
Myszkowski P. B., Skowronski M. E., Olech L., Oslizlo K.,
Hybrid Ant Colony Optimization in solving Multi-Skill Resource-Constrained Project Scheduling Problem,
Soft Computing, DOI: DOI 10.1007/s00500-014-1455-x
============================================================
General characteristics:
Tasks: 20
Resources: 3
Precedence relations: 9
Number of skill types: 3
============================================================
ResourceID       Salary          Skills
1                       0.0             Q2: 2    Q3: 3
2                       0.0             Q2: 1
3                       0.0             Q0: 2
============================================================
TaskID   Duration        Skill   Predecessor IDs
1                       13      Q0: 2
2                       11      Q2: 1
3                       13      Q0: 2
4                       17      Q2: 2
5                       22      Q2: 1           4
6                       10      Q3: 3
7                       12      Q2: 2
8                       27      Q2: 2           7
9                       29      Q2: 1
10                      30      Q0: 2
11                      24      Q2: 2           4
12                      14      Q0: 2
13                      25      Q3: 3
14                      28      Q3: 3
15                      19      Q3: 3
16                      18      Q0: 2
17                      14      Q3: 3           16      12      10
18                      21      Q3: 3           7
19                      13      Q0: 2           5
20                      24      Q3: 3           4
============================================================
```

Figure 7.2.1: The sample of project data file

7.3 Appendix C. Parameter analysis for small instances 3

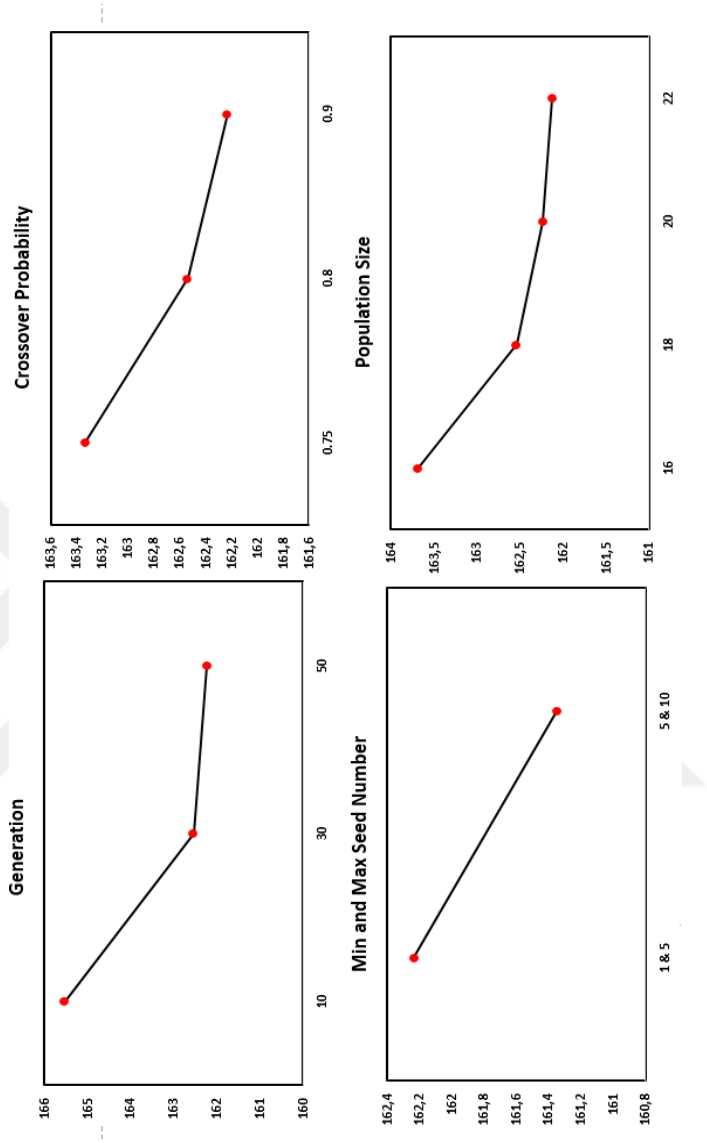


Figure 7.3.1: Parameter analysis for small instance 3

7.4 Appendix D. Parameter analysis for small instance 10

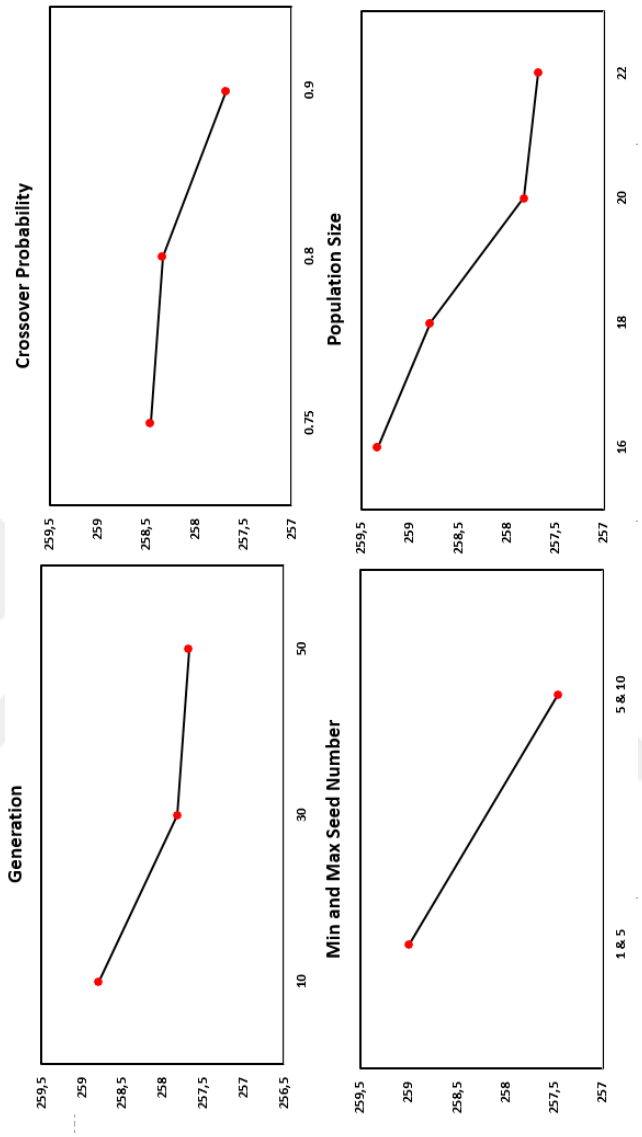


Figure 7.4.1: Parameter analysis for small instance 10

7.5 Appendix D. Parameter analysis for small instance 1

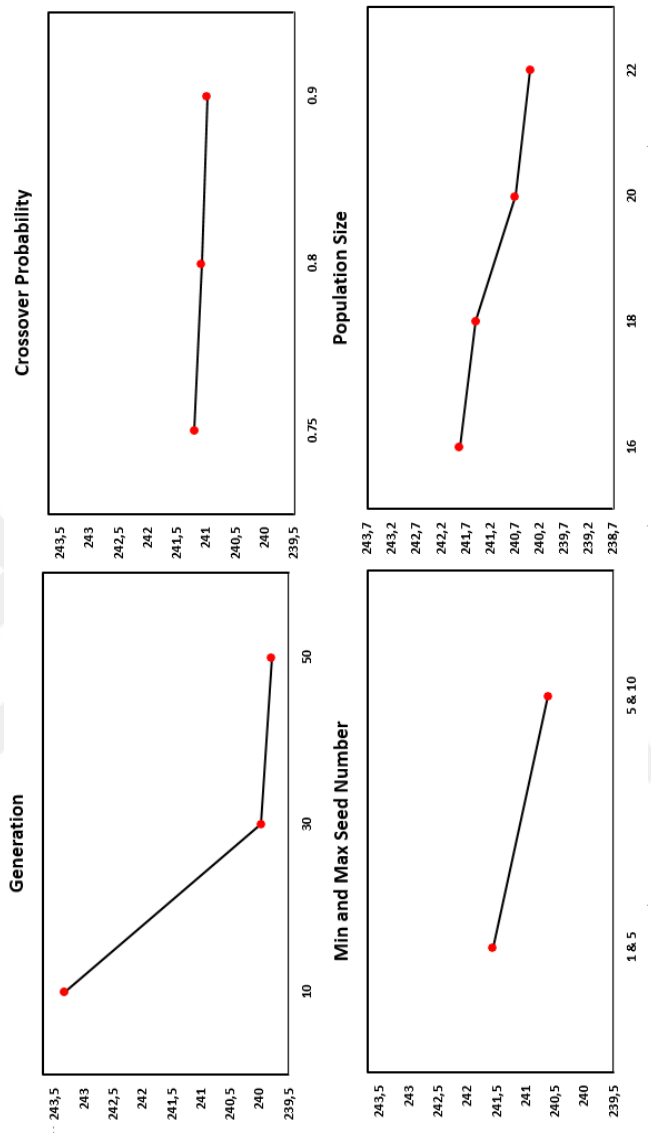


Figure 7.5.1: Parameter analysis for small instance 1

7.6 Appendix E. Parameter analysis for medium instances 18

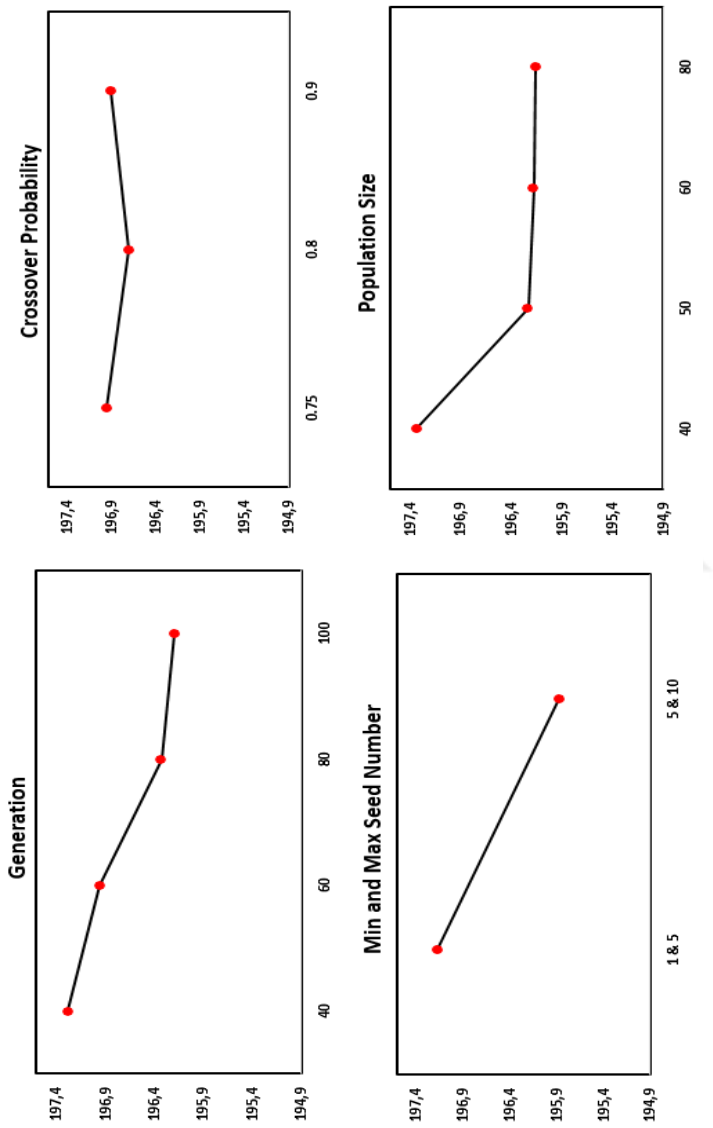


Figure 7.6.1: Parameter analysis for medium instance 18

7.7 Appendix F. Parameter analysis for medium instance 19

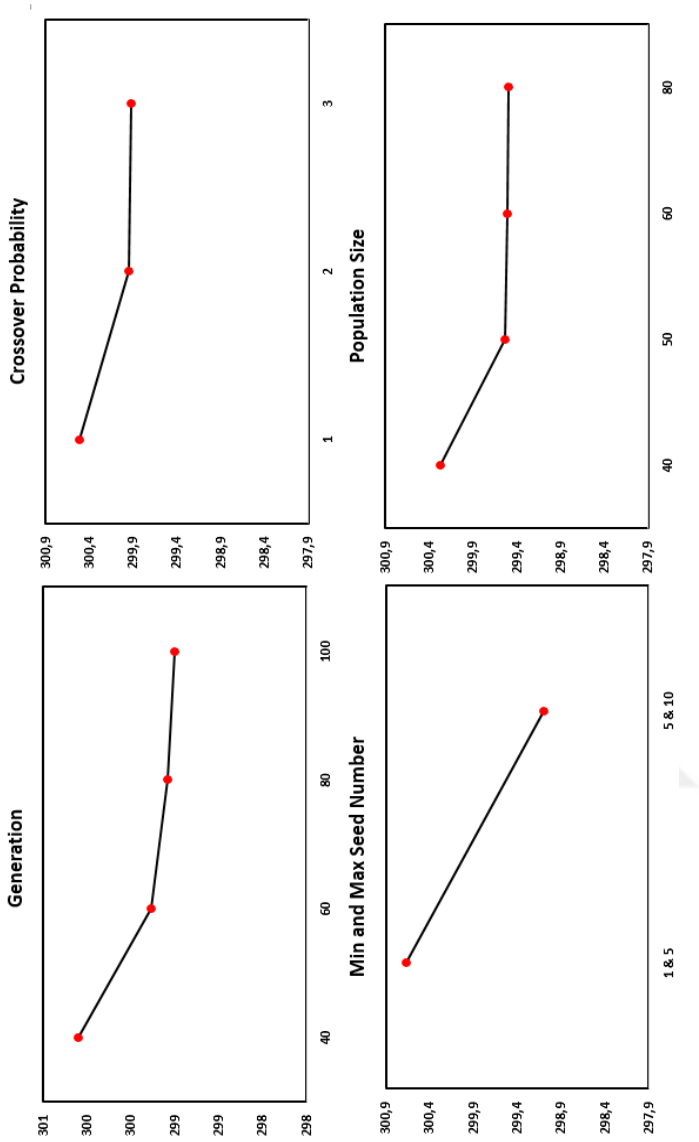


Figure 7.7.1: Parameter analysis for medium instance 19

7.8 Appendix G. Parameter analysis for medium instance 20

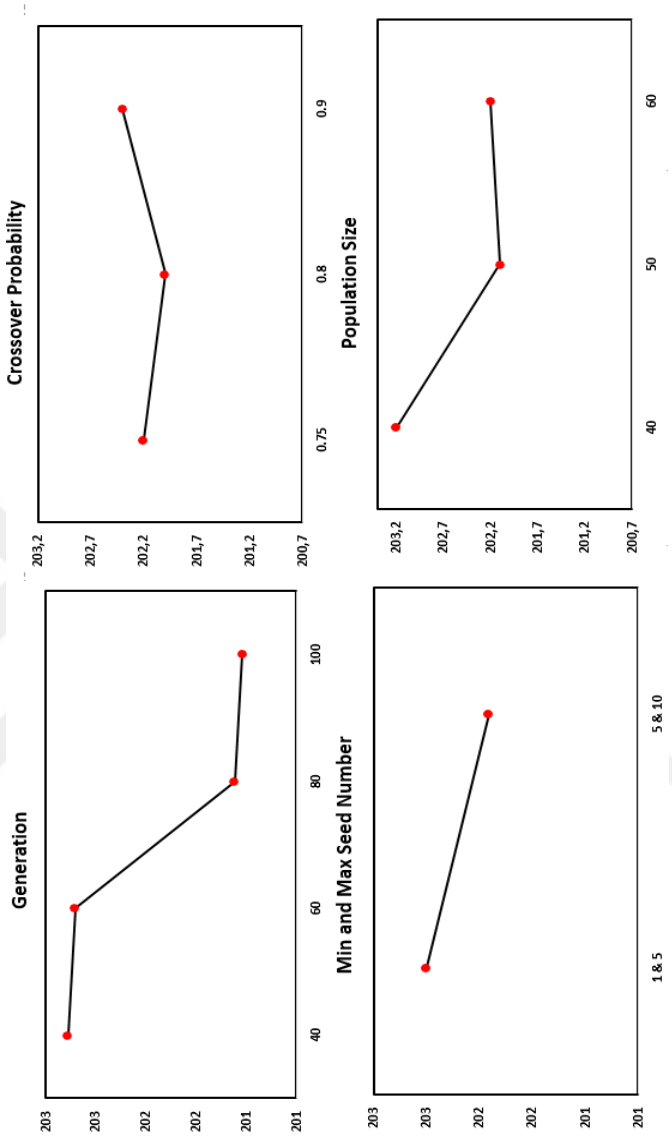


Figure 7.8.1: Parameter analysis for medium instance 20

7.9 Appendix H. Parameter analysis for large instance 21

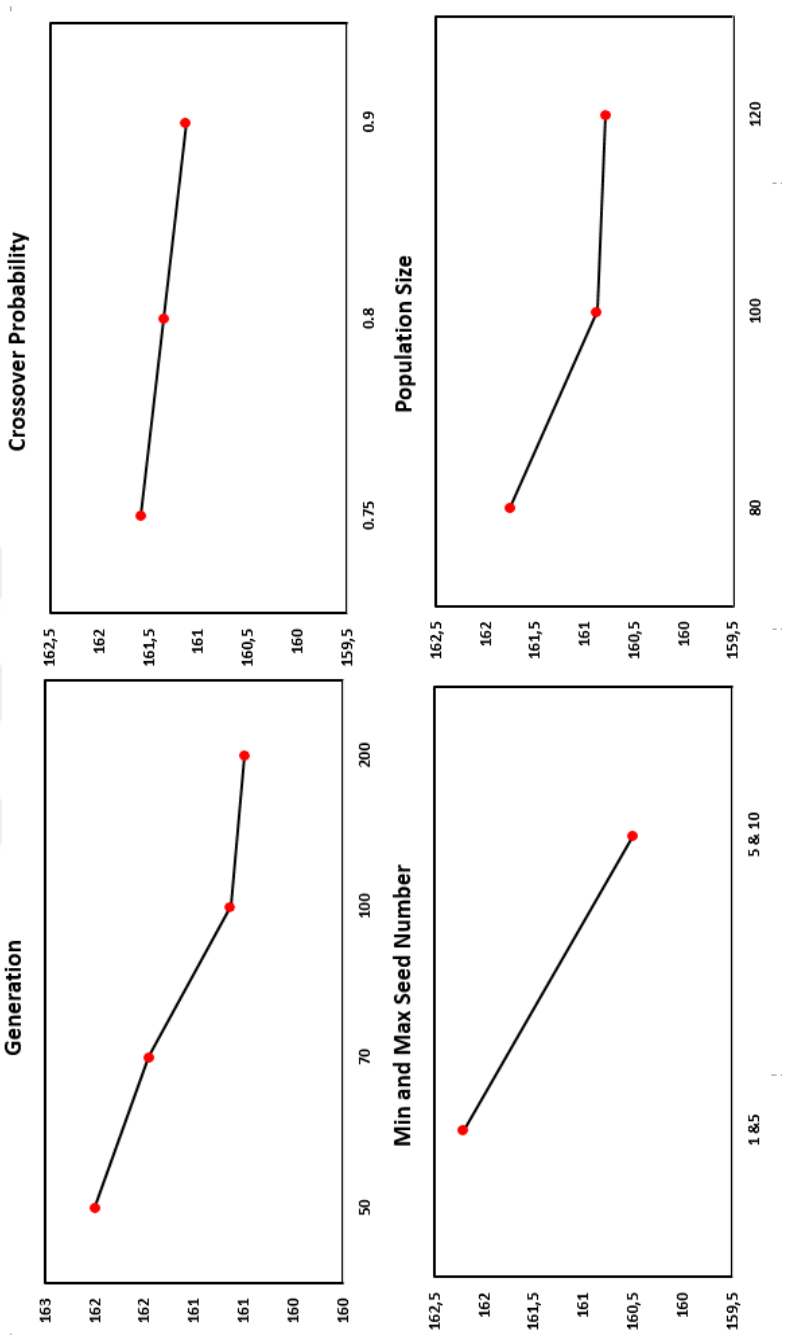


Figure 7.9.1: Parameter analysis l Parameter analysis

7.10 Appendix I. Parameter analysis for large instance 22

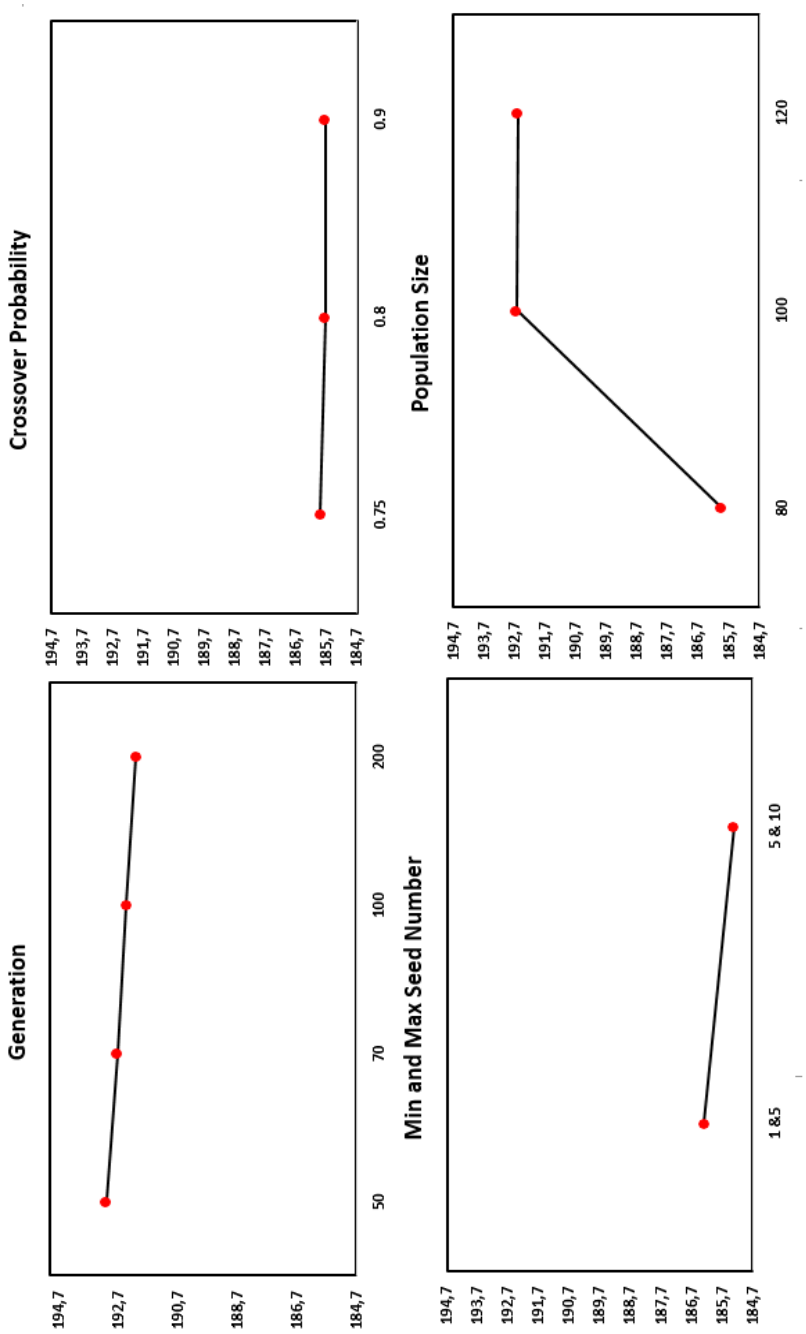


Figure 7.10.1: Parameter analysis for large instance 22

7.11 Appendix J. Parameter analysis for large instance 23

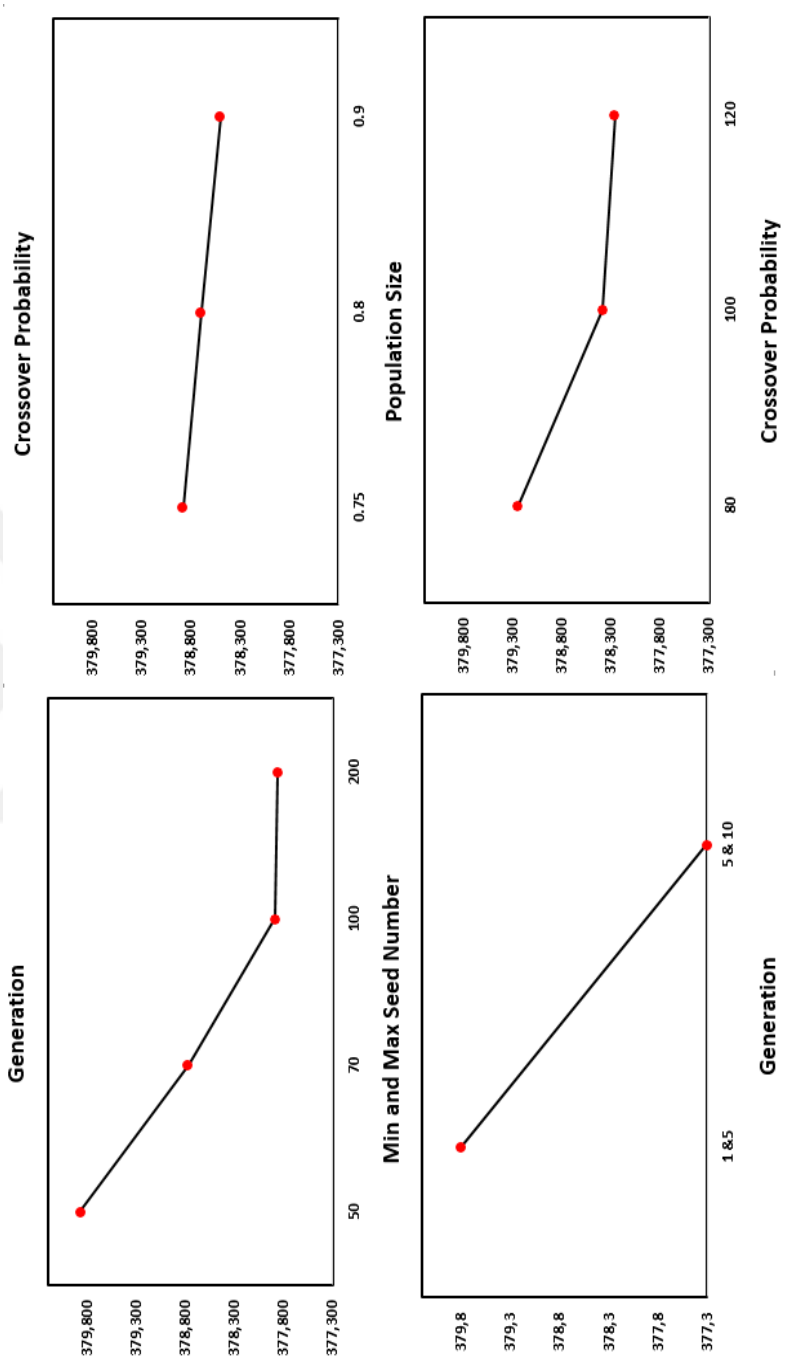


Figure 7.11: Parameter analysis for large instance 23

7.12 Appendix K. Parameter analysis for large instance 27

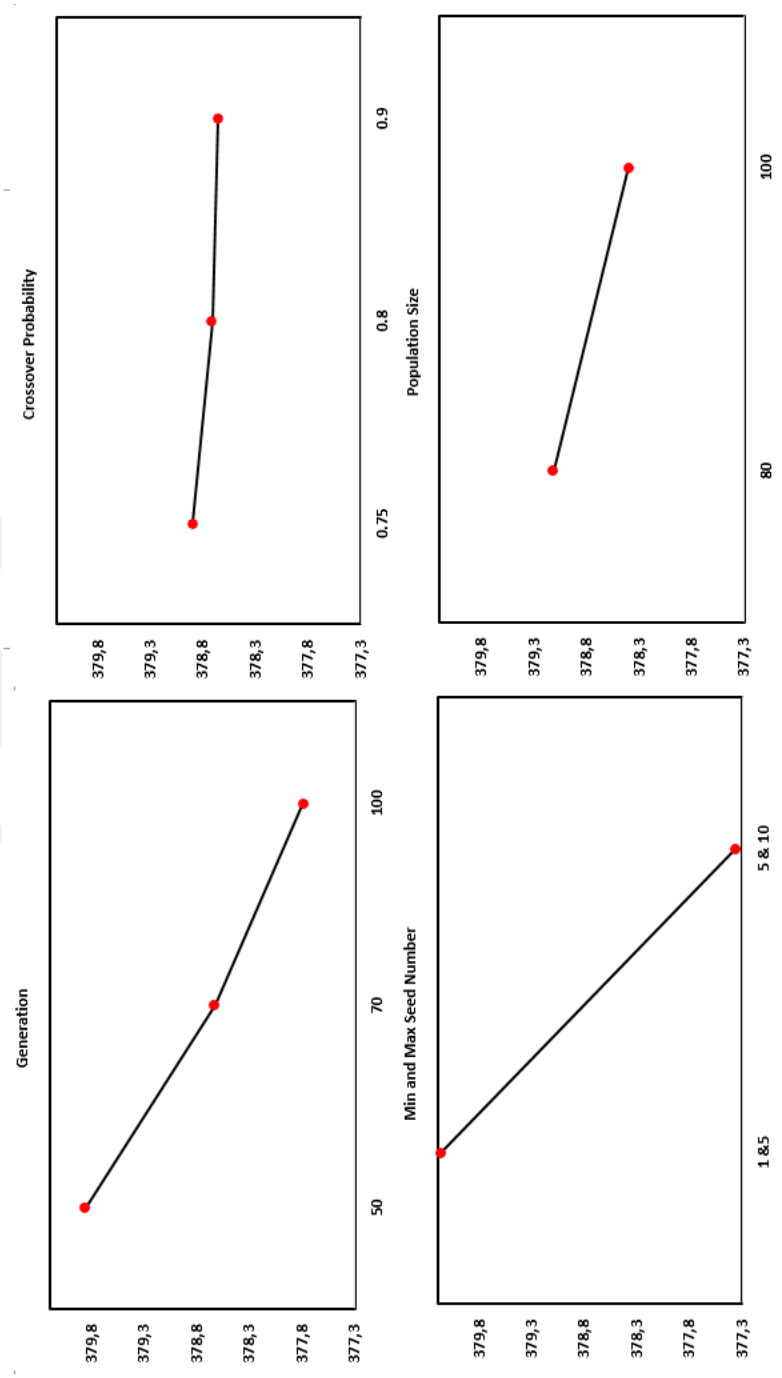


Figure 7.12: Parameter analysis for large instance 27

7.13 Appendix L. Parameter analysis for large instance 24

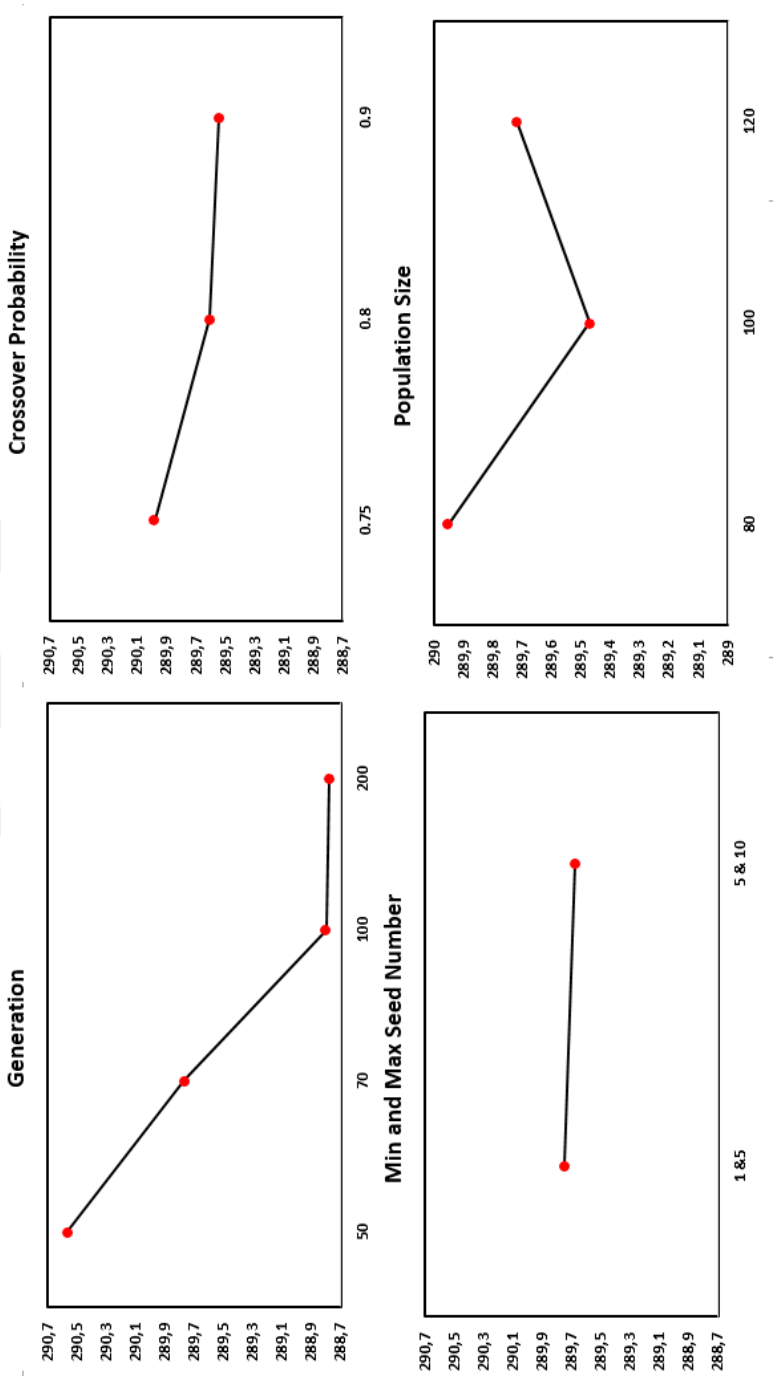


Figure 7.13: Parameter analysis for large instance 24