MASTERTHESIS

Implementation and Analysis of an algorithm portfolio approach for the RCPSP

A Thesis Submitted to:

First Reviewer:
Prof. Dr. Jürgen Zimmermann
Institute of Management and Economics, Operations Research Group
Clausthal University of Technology
Second Reviewer:
Prof. Dr. Jörg Philipp Müller
Institute of Informatics
Clausthal University of Technology
Supervisor:
Lena Sophie Wohlert, M.Sc.

by:
Franklin Leugue
Rollstrasse 40
38678 Clausthal-Zellerfeld
Business Informatics, 9. Semester
Matriculation number: 515454

Academic Year: Winter 2023/2024

February 3, 2024

Contents

Acronym

$RCPSP$ Resource Constrained Project Scheduling Problem

$COPs$ Combinational Optimization Problems

$ASPs$ Algorithm Selection Problems

MPM Metra Potential Method

$PSPLIB$ Project Scheduling Problem Library

SVM Support Vector Machine

Symbol

$A(S,t)$ Set of activities in execution at time t for given schedule S.

D Mean duration of all activities in the instance

E Set of arrows

f Fitness function

p_i duration of activity i

$P(t)$ Population at time t

N node

NA number of activities in the instance

$R_K(S,t)$ The amount of resource k that is needed at time t is required to carry out the activity $i \in A(t)$

R_k represents the availability of each resource type k in each time period

r_{ik} Resource requirement for activity i of resource k

S_i starting time of activity i for the execution

S Set of activities

s number of successors

(V,E) Priority network

List of Figures

List of Tables

Zusammenfassung

Das Problem der ressourcenbeschränkten Projektplanung (RCPSP) ist ein gut untersuchtes kombinatorisches Optimierungsproblem. In der Literatur gibt es eine Reihe von etablierten Heuristiken und experimentelle Studien haben gezeigt, dass je nach Problemstellung die eine oder andere Methode manchmal bessere Lösungen liefert. Die Masterarbeit beschäftigt sich mit der Frage, ob die automatische Auswahl von Algorithmen die Leistung im Vergleich zur Verwendung eines einzelnen Algorithmus für das ressourcenbeschränkte Projektplanungsproblem verbessern kann. Es soll untersucht werden, welche Heuristiken abhängig von den Eigenschaften der Instanz mehr oder weniger effektiv sind. Zur Lösung des Problems wird zunächst ein Portfolio von Algorithmen implementiert. Dieses Portfolio besteht aus genetischen Algorithmen mit unterschiedlichen Darstellungen wie Aktivitätslisten und Zufallsschlüsseln. Anschließend werden verschiedene Konfigurationen unter Verwendung einer der Repräsentationen für die Implementierung definiert und es wird untersucht, ob sich die Algorithmen gegenseitig ergänzen, um gute Lösungen für eine große Anzahl von Instanzen zu finden. Auf dieser Grundlage wird eine automatische Algorithmenauswahl mit Hilfe von maschinellem Lernen entworfen, implementiert und evaluiert.
Ergebnisse für eine Reihe von Instanzen einer Fallstudie mit einer großen Anzahl von Aktivitäten unter Verwendung von vier definierten Konfigurationen oder Algorithmen für die Berechnung erstellt umfassen zwei Formen der Darstellung der Lösung, und Austausch von Parametern des generischen Algorithmus, kamen wir zu dem Schluss, dass bis zu diesem Punkt wurden unterschiedliche Ergebnisse geliefert, und dass die Form der Darstellung der Lösung oder die Parameter des genetischen Algorithmus kann das Ergebnis der Instanz in Bezug auf die Variation der besten Ergebnisse zu beeinflussen. Einige der im Portfolio enthaltenen Algorithmen lieferten zufriedenstellende Ergebnisse und wir mussten zum maschinellen Lernen übergehen, indem wir die überwachten Lernmodelle anwendeten und den ausgewählten Entscheidungsbaum-Modus trainierten. Das Ergebnis dieses Trainings lieferte uns vielversprechende Ergebnisse für zwei verglichene Konfigurationen in Bezug auf die mittlere Zeit und die mittlere Zeitvorhersage des Modells, was zeigt, dass eine automatische Auswahl des Algorithmus besser ist als nur ein

einziger Algorithmus oder eine einzige Konfiguration, die zur Lösung eines Problemfalls erstellt wurde.

Schlüsselwörter - RCPSP, Algorithmusportfolio - Genetischer Algorithmus - Aktivitätsliste - Zufallsschlüssel - Maschinelles Lernen

Abstract

The Problem of Resource Constrained Project Scheduling (RCPSP) is a well-studied combinatorial optimisation problem. There are a number of well-established heuristics in the literature, and experimental studies have shown that one method or the other sometimes provides better solutions, depending on the problem instance. The question of the Master's thesis is whether automatic algorithm selection can improve performance compared to using a single algorithm for Resource Constrained Project Scheduling Problem. The aim is to study which heuristics are more or less effective depending on the properties of the instance, and to solve this problem a portfolio of algorithms will be implemented first. This portfolio will include genetic algorithms with different representations of solution, such as activity lists and random keys. Then, different configurations using one of the representation forms of solution will be defined for implementation, and it will be determined whether the algorithms complement each other to find good solutions for a large number of instances. Based on this, an automatic algorithm selection using machine learning will be designed, implemented and evaluated .
Solutions of a set of instances of a case study with a large number of activities using four defined configurations or algorithms created for the computation include two forms of solution representations, and swapping parameters of the generic algorithm, we came to the conclusion that up to this point different results were provided, and that the representation form of solution or parameters of Genetic algorithm can affect the result of instance regarding the variation of best results provided, some of the algorithms contained in the portfolio showed satisfyed result and it was necessary for us to come to the machine learning approach by applying the supervised learning models and training the selected decision tree mode. The output of this training gave us promising results for two compared configurations in terms of its mean time and the mean time prediction of the model, showing that an automatic selection of the algorithm is better than having only a single algorithm or confguration created to solve a problem instance.
Keywords - RCPSP, Algorithm portfolio, Genetic Algorithm, Activity-list, Random-Vector, Machine Learning

Chapter 1

Contextualizing the problem

1.1 Introduction

Previous solutions to the problem of scheduling projects with limited resources using a single algorithm were based on the search for an optimal or near-optimal solution but seemed to have made little positive contribution. In the following years, the concept of algorithm portfolios emerged as a promising approach to improve the quality and robustness of solutions by combining several genetic algorithms. Various variants of genetic algorithms have been developed over the years to solve resource-constrained project scheduling problems, where resources are renewable and activities are executed in a very precise manner. However, thanks to artificial intelligence through machine learning, we have been able to optimize the results of solving resource-constrained project scheduling problems. This is the goal of our research, which aims to use machine learning to select the best algorithm from the portfolio of genetic algorithms for solving a given instance problem, in order to optimize the result by minimizing its completion time. The aim of our work will therefore be to use these foundations to develop a portfolio of algorithms, based on two representation of the results, namely the list of activities and the random keys, that can effectively solve the problem of planning a project with limited resources and provide high-quality solutions. Our portfolio will consider different configurations to solve a specific case study where our analysis will be based on the results obtained. In The portfolio thus designed will be integrated into software for the resource-constrained project scheduling problem to minimize the execution time of a given instance problem [20]. To assess the performance of an algorithm portfolio, computational experiments will be performed on a benchmark dataset of resource-constrained project scheduling problems. The experiments will compare the performance of the portfolio individually in the form of configurations exported with different parameters such as solution quality, and

computation time. The analysis and interpretation phase will consist of studying the experimental results for a case study j120 of instances and using the decision tree model at the end to determine whether instances of the same class give consistent results for a given configuration, including different parameters of the genetic algorithm, or whether the results require an automatic selection of algorithms between two defined configurations present in the portfolio, taking into account one or the two forms of representation of the solutions. The results will be interpreted to answer this question. To reach this goal we will start by understanding the concept of the Resource Constrained Project Scheduling Problem and explain all its particular aspects in the first chapters our work after describing the existing literature on algorithmic portfolios and their application to optimization problems such as the resource-constrained project scheduling problem. The next part will focus on the genetic algorithm approach and its parameters and last part will refer to the automatic selection of algorithm.

1.2 Background and Motivation

This study aims to explore the potential advantages of automatic algorithm selection by the application of the machine Learning concept in solving resource-constrained project scheduling problems. The central research question revolves around whether the performance of solving this problem can be enhanced by dynamically selecting the most suitable algorithm for a given problem instance, rather than relying on a single algorithm. This investigation aims to discern the intricate relationship between problem instance properties and the performance of diverse heuristics. Structuring a range of instance characteristics, such as project size, resource availability, and activity dependencies.

To achieve these objectives, this study proposes the implementation of an algorithm portfolio encompassing genetic algorithms with two-form solution representations, such as an activity list and random keys, each with its own strengths and weaknesses. To do this, we will aim to design a portfolio containing several user-adapted configurations or algorithms, each of which will use one of the two representation forms of coding and also take into account the different parameters of the genetic algorithm. This portfolio will exhibit complementary behavior, ensuring that the different algorithms excel in different types of instances. In addition, this thesis aims to exploit machine learning techniques to forge an automatic algorithm selection mechanism based on these defined configurations or algorithms building the portfolio. A machine learning model will be trained on a dataset comprising problem instances of study case j120 and the corresponding performance of algorithms. The ultimate goal of this study is to demonstrate the potential performance improvement of dynamically selecting the most appropriate algorithm

for a given problem instance. By exploiting the strengths of different algorithms for distinct instance types, this approach should lead to superior solutions across a broader spectrum, contributing to more efficient project planning and resource allocation in real-world scenarios.[39]

The experimental design defines a diverse set of problem instances that encapsulate the various characteristics and complexities of the Resource-Constrained Project Scheduling Problem. Performance metrics, such as makespan and computation time of instances, will be specified to assess solution quality and computational efficiency. The automated approach will be implemented using Python programming language within the Visual Studio Code environment. A series of experiments with defined problem instances will be conducted.

1.3 Literature Overview

Project scheduling has become increasingly popular in the make-to-order production industry, and has been applied in various fields. The main focus of these projects is to solve the problem of project scheduling under resource constraints, which has been extensively researched using genetic algorithms. Researchers have implemented several algorithms as part of the genetic algorithm to minimize the time required to complete the project. They have also studied various operators contained in the genetic algorithm, such as selection, crossover, and mutation operators, which can significantly impact the results generated by the genetic algorithm. In this chapter, we conduct a comprehensive analysis of the relevant literature, highlighting key terms, methodologies, and advances in the field.
In his article, Klimek[20] explained a genetic algorithm for scheduling projects with resource constraints;genetic algorithm is defined as a technique inspired by biological evolution.[20] it aims to identify effective solutions and embodies the concept of inheritance, wherein only the fittest individuals endure for participation in reproduction through a combination of crossover and mutation.[20] In the context of scheduling, solutions represented by chromosomes go through evaluation using a fitness function, reflecting the extent to which individuals adapt to their environment. The genetic algorithm features the transmission of genetic information to offspring by the most robust individuals, achieved through crossover and mutation operations. Notably, various selection, crossover, and mutation operators were introduced,with the selection operator defined as a genetic mechanism that picks individuals from the current population for inclusion in the subsequent generation. Similarly, the crossover operator serves as a genetic mechanism merging two parent individuals to generate a new offspring individual.[20] The process of genetic algorithm reflect natural genetic inheritance and the phenomenon of natural selection, where only the strongest individuals survive to participate in reproduction through

crossbreeding and mutation. In the context of scheduling, the evaluation of solutions through the fitness function is necessary. The genetic algorithm also involves the transmission of genetic information to descendants by the strongest individuals through crossover and mutation operations. Marcin Klimek cited three main crossover operators, which are the most widely used and described.[21] The author presented three genetic operators, namely selection, crossover, and mutation operators. The selection operator is responsible for selecting individuals from the current population to include them in the next generation.[20] The crossover operator combines two parent individuals to produce a new child individual. Among these crossover operators, three main operators are cited, which are the most widely used and described by the author because they satisfy precedence constraints. The mutation operator modifies the value of one or more genes in an individual, thus preventing the population from stagnating at local optima. The chromosomal or genome representation used was an activity list with decoding based on a parallel or serial generation scheme. In his article, Vicente Valls [41] proposes another form of genetic algorithm with the name hybrid genetic algorithm. The hybrid algorithm introduces several changes to the GA paradigm, including a new way of selecting the parents to be combined, and a two-phase strategy in which the second phase restarts the evolution from a population close to the best program found in the first phase. The results generated through the calculations highlight that the hybrid genetic algorithm offers advantages in solving instance problems while outperforming other cutting-edge algorithms and offers a great advantage in terms of speed, which can be explained as follows. First, we show that the double justification, front-back improvement, or front-back ordering [22] contained in the genetic algorithm makes it possible to obtain better results.The form of activity representation is a vector of lists of activities. There are different crossover operators presented by authors depending on various aspects of the genetic algorithm such as one-point crossover, two-point crossover, and others that we will clarify during our study depending on the objectives to be achieved. In his article specializing in the study of the selection of combinatorial search algorithms, that is, the selection of the best algorithm to solve a given instance problem, Kotthof Lars [24] describes this method as making it possible to find the most suitable algorithm for a given problem, rather than having to develop numerous algorithms that have limitations because of the heuristics used or the execution environment. This new technique for solving an instance problem through the choice of algorithm offers many advantages, including improved performance of the systems designed. Given the fact that, according to many researchers in this field, a single algorithm cannot give the best performance for all the instances of the problem which it is desired to solve, this was justified by the authors Aha (1992) [1] and Wolpert and Macready (1997) [44], Rice (1976) [37] describes in his publication the problem of selecting

algorithms that aim to match an instance to the algorithm required to solve it, with the help of this correspondence it becomes easy to select a given instance problem the algorithm required to solve it. The recent approach emphasized by authors such as SATzilla (Xu et al. 2008) [45], ArgoSmart (Nikolic, Maric, and Janicic 2009) [32], SALSA (Demmel et al. 2005) [11] and Eureka (Cook and Varnell 1997) [10], which consists of selecting a single algorithm from the portfolio and using it for a complete resolution of an instance problem, is limited to a poor choice, i.e. the wrong or weak algorithm solving the given instance problem. This approach can be used before the resolution of the instance or offline or during the resolution of the instance or online. The author Lars Kotthoff distinguishes between two types of portfolios containing algorithms, static and dynamic portfolios, which can solve given instances. This approach can be used before the instance is solved (offline) or during its resolution. Tommy Messelis and Patrick De Causmaecker, in their article on this subject in the context of a multi-mode system, present a research paper based on the notion of empirical hardness models, which describe models that match the characteristics of the problem instance with the performance of an algorithm in order to predict the performance of a set of algorithms. These predictions can then be used to automatically select the best performing algorithm, taking into account the available computing resources. They describe the steps to be taken in the construction of the automatic selection tool, based on empirical hardness models.[30] This is the basis of our work, in which, after illustrating the problem of limited project resources and the application of the genetic algorithm, and after observing all the parameters involved, including the construction of the genetic algorithm, we seek to optimise our solutions in the context of case of class including a big number of activities, based on the concept of automatic selection using machine learning by training efficient prediction.In the concept of machine learning, the author Salim Gridi presents different prediction models, including supervised learning models[13]. In his article specializing in the study of the selection of combinatorial search algorithms, that is, the selection of the best algorithm to solve a given instance problem, Kotthof Lars [24] describes this method as making it possible to find the most suitable algorithm for a given problem, rather than having to develop numerous algorithms that have limitations because of the heuristics used or the execution environment. This new technique for solving an instance problem through the choice of algorithm offers many advantages, including improved performance of the systems designed. Given the fact that, according to many researchers in this field, a single algorithm cannot give the best performance for all the instances of the problem which it is desired to solve, this was justified by the authors Aha (1992) [1] and Wolpert and Macready (1997) [44], Rice (1976) [37] describes in his publication the problem of selecting algorithms that aim to match an instance to the algorithm required to solve it, with

the help of this correspondence it becomes easy to select a given instance problem the algorithm required to solve it. The recent approach emphasized by authors such as SATzilla (Xu et al. 2008) [45], ArgoSmart (Nikolic, Maric, and Janicic 2009) [32], SALSA (Demmel et al. 2005) [11] and Eureka (Cook and Varnell 1997) [10], which consists of selecting a single algorithm from the portfolio and using it for a complete resolution of an instance problem, is limited to a poor choice, i.e. the wrong or weak algorithm solving the given instance problem. This approach can be used before the resolution of the instance or offline or during the resolution of the instance or online. The author Lars Kotthoff distinguishes between two types of portfolios containing algorithms, static and dynamic portfolios, which can solve given instances. This approach can be used before the instance is solved (offline) or during its resolution. Tommy Messelis and Patrick De Causmaecker, in their article on this subject in the context of a multi-mode system, present a research paper based on the notion of empirical hardness models, which describe models that match the characteristics of the problem instance with the performance of an algorithm in order to predict the performance of a set of algorithms. These predictions can then be used to automatically select the best performing algorithm, taking into account the available computing resources. They describe the steps to be taken in the construction of the automatic selection tool, based on empirical hardness models.[30] This is the basis of our work, in which, after illustrating the problem of limited project resources and the application of the genetic algorithm, and after observing all the parameters involved, including the construction of the genetic algorithm, we seek to optimise our solutions in the context of case of class including a big number of activities, based on the concept of automatic selection using machine learning by training efficient prediction.In the concept of machine learning, the author Salim Gridi presents different prediction models, including supervised learning models[13]. In his article specializing in the study of the selection of combinatorial search algorithms, that is, the selection of the best algorithm to solve a given instance problem, Kotthof Lars [24] describes this method as making it possible to find the most suitable algorithm for a given problem, rather than having to develop numerous algorithms that have limitations because of the heuristics used or the execution environment. This new technique for solving an instance problem through the choice of algorithm offers many advantages, including improved performance of the systems designed. Given the fact that, according to many researchers in this field, a single algorithm cannot give the best performance for all the instances of the problem which it is desired to solve, this was justified by the authors Aha (1992) [1] and Wolpert and Macready (1997) [44], Rice (1976) [37] describes in his publication the problem of selecting algorithms that aim to match an instance to the algorithm required to solve it, with the help of this correspondence it becomes easy to select a given instance problem

the algorithm required to solve it. The recent approach emphasized by authors such as SATzilla (Xu et al. 2008) [45], ArgoSmart (Nikolic, Maric, and Janicic 2009) [32], SALSA (Demmel et al. 2005) [11] and Eureka (Cook and Varnell 1997) [10], which consists of selecting a single algorithm from the portfolio and using it for a complete resolution of an instance problem, is limited to a poor choice, i.e. the wrong or weak algorithm solving the given instance problem. This approach can be used before the resolution of the instance or offline or during the resolution of the instance or online. The author Lars Kotthoff distinguishes between two types of portfolios containing algorithms, static and dynamic portfolios, which can solve given instances. This approach can be used before the instance is solved (offline) or during its resolution. Tommy Messelis and Patrick De Causmaecker, in their article on this subject in the context of a multi-mode system, present a research paper based on the notion of empirical hardness models, which describe models that match the characteristics of the problem instance with the performance of an algorithm in order to predict the performance of a set of algorithms. These predictions can then be used to automatically select the best performing algorithm, taking into account the available computing resources. They describe the steps to be taken in the construction of the automatic selection tool, based on empirical hardness models.[30] This is the basis of our work, in which, after illustrating the problem of limited project resources and the application of the genetic algorithm, and after observing all the parameters involved, including the construction of the genetic algorithm, we seek to optimise our solutions in the context of case of instance class including a big number of activities, based on the concept of automatic selection using machine learning by training efficient prediction.In the concept of machine learning, the author Salim Gridi presents different prediction models, including supervised learning models[13].

Chapter 2

Problem Description

To better understand the purpose of the research it is necessary to describe and clarify the Resource Constrained Project Scheduling Problem presenting for genetic algorithms. After that, an overview of the algorithm selection will be very important.

2.1 Description of the Resource-Constrained Project Scheduling Problem

Project scheduling, as an integral part of project management, involves determining the optimal time to start or finish activities within a project.All activities must adhere to priority and resource constraints while optimizing specific criteria.

2.1.1 Introduction of Identifiers

In this section, we present the key identifiers that form the basis of the analysis. These include:

- the set of activities V, all the sets of activities necessary for the project termination.
- execution times p_i of each activity i, the time required for an activity
- start times S_i with Schedule S;
- Relationships of priority that constrain the time schedule.

2.1.2 Resources, Resource Uses, and Capacities

Following the introduction of identifiers, we examine aspects related to resources. This includes:

- Definition of resources
- use of resources associated with each activity, notified by r_ik which indicates the quantity of resource k required at time t for executing activities of type k
- Resource capacities that impose constraints on resource allocation notified by R_k.

2.2 Problem Formulation

The priority network is represented by (V, E), where the nodes N represent activities and the arcs E represent priority relationships.[2] A project consists of $V = n + 2$ activities, each of which requires uninterrupted processing. Activities are represented by identifiers j with durations p_j and are subject to priority relationships and resource constraints. Dummy activities 0 and $n + 1$ have durations of zero and serve as initial and final activities or as the beginning and end of the project, i.e. as predecessors and successors of all other activities.[2] In the case of the deterministic RCPSP, where each duration is a constant, the objective of solving the RCPSP is to establish a schedule S, denoted by start times $(s_1, s_2, ..., s_n)$, that minimizes the makespan,[20], the total duration of the project realization while adhering to both precedence and resource constraints. Reducing the completion time of the distinctive dummy ending activity is essential to minimize the total duration of the project.[46].

2.2.1 MPM Representation

The Metra Potential Method (MPM) is used for graphical representation, where nodes correspond to activities and arcs denote temporal relationships.[47] It helps visualize the project schedule and relationships between activities.

Example:

We provide a concrete example that illustrates the Metra Potential Method representation for a given project scenario. The example in Figure 2.1 serves as a reference point for grasping the application of the Metra Potential Method.

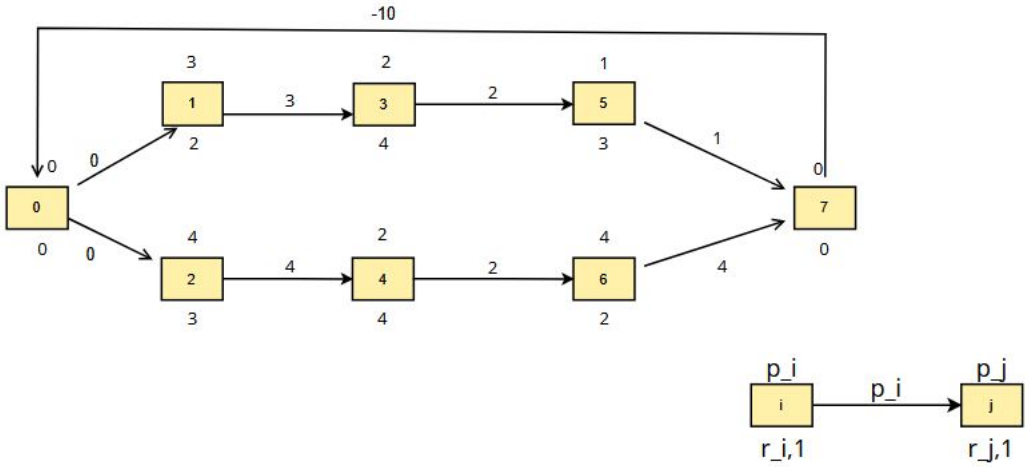

Figure 2.1: Example of a project network based on the MPM Netzplan representation.

The figure shows an example of a project network using the Metra Potential Method representation, including a renewable resource type ($K = 1$) and specifying a resource capacity ($R_1 = 4$). The project consists of 7 activities, where activities 0 and 7 are dummy or fictitious activities and activities 1 to 6 are real activities that have to be scheduled [17]. Each activity is assigned the earliest start time.

2.2.2 Calculation of ES and LS

To further explain the scheduling process, we explain how to determine the early start (ES) and late start (LS) times for activities.[4] Understanding these critical times is fundamental to creating a feasible project schedule.

Methodology:

We outline the methodology used to calculate ES and LS for each activity, taking into account priority relationships and resource constraints using the triple algorithm. The algorithm is described as follows [47]:

Application:

The use of the triple algorithm makes it possible to calculate the earliest and latest start times for each activity, which are displayed in a distance matrix table D. **Initialization:** Set $d_{ii} := 0$ and $p_{ii} := i$ for all activities $i \in V = \{0, \ldots, 7\}$.

Algorithm 1 Triple Algorithm

Initialization:
for $i \in V$ **do**
  Set $d_{ii} := 0$ and $p_{ii} := i$.
end for
for $j \in V \setminus \{i\}$ **do**
  Set $d_{ij} := \begin{cases} \delta_{ij}, & \text{if } \{i,j\} \in E, \\ -\infty, & \text{otherwise} \end{cases}$
  Set $p_{ij} := \begin{cases} i, & \text{if } d_{ij} > -\infty, \\ -1, & \text{otherwise} \end{cases}$
end for
Main step:
for $\nu \in V$ **do**
  for $i \in V \setminus \{\nu\}$ **do**
    for $j \in V \setminus \{\nu\}$ **do**
      Set $d_{ij} := \max(d_{ij}, d_{i\nu} + d_{\nu j})$ and $p_{ij} := (d_{ij} > d_{i\nu} + d_{\nu j})?p_{\nu j} : p_{ij}$.
    end for
  end for
end for

Set $d_{ij} := p_{ij}$ and $p_{ij} := i$ for all $< i, j > \in E$; for other cases, $d_{ij} := -\infty$ and $p_{ij} := -1$.

After using the triple algorithm for the first step $v := 0$, we get the following results for the figures 2.2 and 2.3:

j / i	0		1		2		3		4		5		6		7	
	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}
0	0	0	0	0	0	0	-∞	-1	-∞	-1	-∞	-1	-∞	-1	-∞	-1
1	-∞	-1	0	1	-∞	-1	3	1	-∞	-1	-∞	-1	-∞	-1	-∞	-1
2	-∞	-1	-∞	-1	0	2	-∞	-1	4	2	-∞	-1	-∞	-1	-∞	-1
3	-∞	-1	-∞	-1	-∞	-1	0	3	-∞	-1	2	3	-∞	-1	-∞	-1
4	-∞	-1	-∞	-1	-∞	-1	-∞	-1	0	4	-∞	-1	2	4	-∞	-1
5	-∞	-1	-∞	-1	-∞	-1	-∞	-1	-∞	-1	0	5	-∞	-1	1	5
6	-∞	-1	-∞	-1	-∞	-1	-∞	-1	-∞	-1	-∞	-1	0	6	4	6
7	-10	7	-∞	-1	-∞	-1	-∞	-1	-∞	-1	-∞	-1	-∞	-1	0	7

Figure 2.2: Results First Iteration

If we increase the value of iteration v to $v := 1$, we get the following result:

The final table 2.4 is obtained after running all the iterations up to the last one with $v := 7$:

j / i	0		1		2		3		4		5		6		7	
	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}
0	0	0	0	0	0	0	-∞	-1	-∞	-1	-∞	-1	-∞	-1	-∞	-1
1	-∞	-1	0	1	-∞	-1	3	1	-∞	-1	-∞	-1	-∞	-1	-∞	-1
2	-∞	-1	-∞	-1	0	2	-∞	-1	4	2	-∞	-1	-∞	-1	-∞	-1
3	-∞	-1	-∞	-1	-∞	-1	0	3	-∞	-1	2	3	-∞	-1	-∞	-1
4	-∞	-1	-∞	-1	-∞	-1	-∞	-1	0	4	-∞	-1	2	4	-∞	-1
5	-∞	-1	-∞	-1	-∞	-1	-∞	-1	-∞	-1	0	5	-∞	-1	1	5
6	-∞	-1	-∞	-1	-∞	-1	-∞	-1	-∞	-1	-∞	-1	0	6	4	6
7	-10	7	-10	0	-∞	-1	-∞	-1	-∞	-1	-∞	-1	-∞	-1	0	7

Figure 2.3: Results Second Iteration

j / i	0		1		2		3		4		5		6		7	
	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}
0	0	0	0	0	0	0	3	1	4	2	5	3	6	4	10	6
1	-4	0	0	1	-4	0	3	1	0	2	5	3	2	4	6	5
2	0	0	0	0	0	2	3	1	4	2	5	3	6	4	10	6
3	-7	7	-7	0	-7	0	0	3	-3	2	2	3	-1	4	3	5
4	-4	7	-4	0	-4	0	-1	1	0	4	1	3	2	4	6	6
5	-9	7	-9	0	-9	0	-6	1	-5	2	0	5	-3	4	1	5
6	-6	7	-6	0	-6	0	-3	1	-2	2	-1	3	0	6	4	6
7	-10	7	-10	0	-10	0	-7	1	-6	2	-5	3	-4	4	0	7

Figure 2.4: Results Last Iteration

In this last table on figure 2.4, still called *distanzmatrix*, we can deduce the earliest start time (ES) of each activity from the first row and the latest start time (LS) from the first column. These values can be seen in the table 2.1: above

Table 2.1: Earliest and Latest Starting Time of Activities

$i \in V$	0	1	2	3	4	5	6	7
ES_i	0	0	0	3	4	5	6	10
LS_i	0	4	0	7	4	9	6	10

2.2.3 Mathematical Model of RCPSP

With a clear understanding of the identifiers and resources, we proceed to present the mathematical model that governs the Resource Constrained Project Scheduling Problem. The model should capture the objective function, the decision variables and any constraints imposed by the problem.[20] The Problem can be formulated as an optimisation problem:

$$\text{minimize: } s_{n+1} \tag{1}$$

Subject to:

$$s_i + p_i \leq s_j, \forall (i,j) \in E, \quad s_i \geq 0, \quad s_j \geq 0 \tag{2}$$

$$\sum_{\forall i \in A(t)} r_{ik} \leq R_k, \forall t, \forall k \tag{3}$$

Equation (1) aims to minimise the value of the variable s_{n+1} the duration time of the last activity, (2) represents the priority relations, and inequalities (3) limit the resource utilisation at any time t to the resource capacity R_k. The objective is to find a schedule S of activities, i.e. a set of start times (s1 . . . , sn), where s1 = 0, the priority and resource constraints are satisfied by minimising the makespan or duration of realization of the project. a schedule S of activities, i.e. a set of start times (s1, s2 . . . , sn), where the priority and resource constraints are satisfied by minimising the makespan of the project.[41]

2.2.4 Constraints and Interrelations

The project activities are interrelated by two types of constraints:

- Precedence constraints ensure that each activity j is scheduled after all predecessor activities P_j have been completed.
- Activities require resources with limited capacity.

2.2.5 Schedule Generation Schemes

Schedule generation schemes (SGS) form the basis of many heuristic approaches to the resource-constrained project scheduling problem. These schemes start from scratch, constructing a viable schedule by incrementally extending a partial schedule. A partial schedule is a schedule in which only a subset of the n + 2 activities have been allocated time slots. There are two different types of SGS, which differ in their approach to scheduling. They can be categorised according to their

treatment of activities and time increments. The so-called serial SGS focuses on activity increments, while the parallel SGS focuses on time increments.[22]
Let C be the set of activities already scheduled and $\overline{C}$ the set of activities not yet

Algorithm 2 Serial schedule generation scheme [47]

1: Set $C := \{0\}$ and $S^C := (0)$.
Main step:
2: **for** $\alpha = 1, \ldots, n+1$ **do**
3: Determine $\hat{E} := \{j \in C | Pred(j) \subseteq C\}$.
4: Choose activity $j \in \hat{E}$ with high priority.
5: Determine $ES_j := \max_{i \in Pred(j)}(S_i + \delta_{ij})$.
6: Determine $t^* := \min\{t \geq ES_j | rk(S^C, \tau) + r_{jk} \leq R_k$ for $t \leq \tau < t + p_j$ and all $k \in R\}$.
7: Schedule activity j at time t^*, i.e. set $S_j := t^*$ and $C := C \cup \{j\}$.
8: **end for**
9: **return** S^C

scheduled. We start with SC = (0) with C = $\{0\}$. we also have to specify that the evaluation of the arrow δ_{ij} in the current case is represented by the execution time of the previous notified activity p_i. In each step, we extend the current subschedule SC by an activity j whose predecessor is in the project network. Activity j whose predecessor is already scheduled in the project network. We select j from the set of schedulable activities. $\hat{E} := \{j \in C \mid \text{Pred} \prec D(j) \subseteq C\}$, The number of activities that can be scheduled.If $\hat{E} > 1$, we choose the activity with the largest priority rule value. j is scheduled at the earliest time and resource allowed, i.e. at $t^{:=mintESj(SC)}$. j is scheduled at the earliest possible time and resource, i.e. at $t^* := \min\{t \geq ES_j(SC) \mid rk(SC, \tau) + r_{jk} \leq R_k$ for $t \leq \tau < t + p_j$ and all $k \in R$ [47]. Applying the schedule generation schemes to the example given in Figure 2.1, for the following feasible schedule $(2, 4, 6, 1, 3, 5)$, we obtain the schedule $S = (0, 6, 0, 10, 4, 12, 6, 13)$ of the start times of the project $\{0, 1, 2, 3, 4, 5, 6, 7\}$.

2.2.6 Priority rule

The selection of an activity from the group of plannable activities is based on a priority rule. There are several possible priority rules. In this case, the LST rule was used for the study, where the activity with the smallest Latest Start Time (LST) is selected. We determine the next activity j C to be scheduled according to this priority rule: **LST-rule (Latest Start Time):** Select operation j with the currently smallest latest start time, i.e:
$j := \min\{i \in \hat{C} \mid \min_{h \in \hat{C}} \text{LS}_h(SC)\}$.

2.2.7 Resource Profile:

We also introduce the concept of a resource profile for an eligible program. The resource profile provides insight into resource usage over time and helps to identify potential bottlenecks or areas of resource contention. It can be represented graphically for the given example by the following Gantt diagram 2.5 for the given project instance.

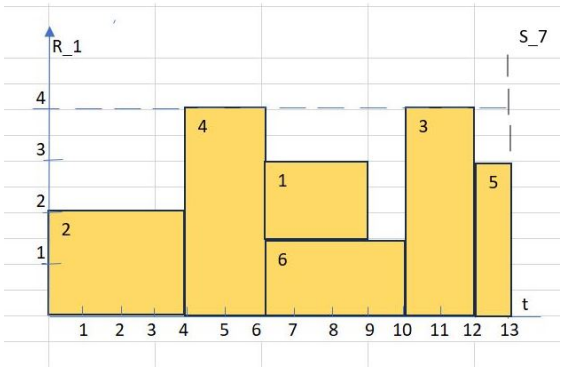

Figure 2.5: Sequence in Gantt Diagram of the given example in Figure 1.

Chapter 3

Genetic Algorithm

Genetic algorithms were specified by Holland in 1975, as metaheuristics, they are multi-point parallel stochastic search algorithms widely used for solving many optimization problems. It is applied to a population of individuals inspired by the principle of biological evolution [14], a class of optimization algorithms that are inspired by the process of natural selection. In genetic algorithms, the fittest individuals are selected to participate in reproduction (crossover, mutation) and produce offspring [20]. Always, the strongest individuals as parents pass on their information to their descendants or children through crossover and mutation operations. Subsequent generations are better adapted to the environment than their original parents[20].
There are several variants of genetic algorithms in the research literature and their applications vary. The focus of our research is on the implementation of genetic algorithms on the basis of two representation forms:

- **The random key based representation form**
- **The activity list based representation form**

In general, genetic algorithms follow a pattern described in the following pseudocode: The application of Genetic algorithm includes relevant parts, which enable the distinction from differents variants of genetic Algorithm [20].

3.1 Description of Elements of the Genetic Algorithm

Each following element graphically presented in the general process (figure 3.1) of the genetic algorithm will be described in this part.

Algorithm 3 Genetic Algorithm [20]

```
1: procedure GA
2:     t := 0;
3:     initialize;
4:     generate P(0); generating initial population
5:     evaluate P(0); fitness function evaluation
6:     while not stop_condition do
7:         t := t + 1;
8:         select x per cent best of P(t-1) to P(t);
9:         create offspring for P(t); crossover, mutation, or other changes
   to create new generation
10:        evaluate P(t); fitness function evaluation for the new generation
11:    end while
12: end procedure
```

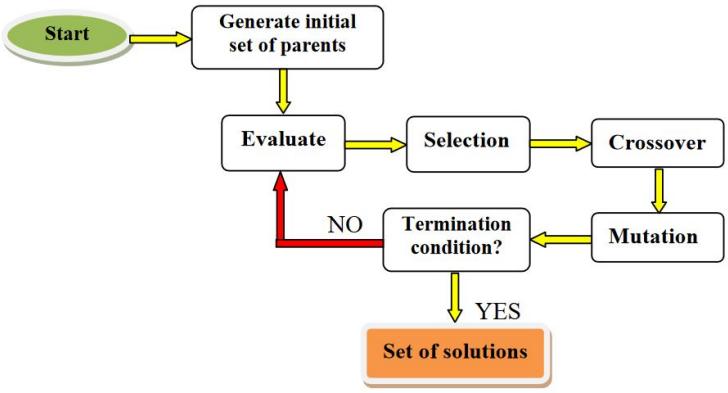

Figure 3.1: The General Process of a Genetic Algorithm [42]

The main points of genetic algorithm are following: [20]

- **coding and method for decoding solutions for a problem**
- **method of generating of initial population**
- **evaluation (fitness) function which measures the quality of the function**
- **methods of selecting individuals for the next generation**
- **crossovers operators**
- **mutation operators**

3.2 Representation/Encoding

This is the form of chromosomal representation of the RCPSP, including the list of activities and the random key representation, the method used to obtain the result can be explained by the encoding.

- **encoding with an activity list:** The solution is encoded in a priority feasible list of activities, where each activity can appear in any position in the list after all of its predecessors.We will sequentially schedule the activities in the order provided by the list, ensuring that each activity is scheduled only after all of its predecessors have been completed (forward scheduling). In addition, each activity is given a priority number, which represents the order according to the earliest possible start time.Using the project instance specified in figure 2.1, we have the list of activities of the feasible schedule, using for each activity its respective index, which explains the order of execution in figure3.2.

- **encoding with random key:**
  In the context of optimization algorithms, each solution is represented as an array of n random keys. A random key is a real number that is randomly generated in the continuous interval [0,1).[38]These numbers are used as sort keys to decode the solution, using the example of the project instance, we can have the following generated random key representation on numbers 3.3 [20]

.

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

Figure 3.2: activity list representation for example of figure 2.1 [17]

1	3	2	5	4	6
0.58	0.64	0.31	0.87	0.09	0.34

Figure 3.3: random key representation for example of figure 2.1 [17]

3.3 decoding

In general, the schedule generation scheme generates a feasible schedule, which is used for the two forms of representation and in the case of the research for the next step with each population generated, the schedule generation scheme begins with an an initial set of unsequenced tasks and constructs a schedule through the incremental expansion of a schedule.[20] In the case of our search, we will use the serial schedule generation scheme form to generate a feasible schedule and test the solution of the given population. At each step, an activity is selected and scheduled. The process ensures that the schedule is completed in finite time and that all capacity constraints are satisfied. The generated schedule S^C is represented by a list of the earliest start times for each activity.
For activity lists and random key based representation forms: The serial schedule generation scheme will play an important role in classical machine scheduling, where it is called list scheduling.[22]

3.4 Initial Population

The initial chromosomes or initial populations can be generated either:

- **randomly:** generation of initial population with random solutions
- **heuristic initialization:** of population using a heuristic for the problem.

3.5 Fitness Function and Selection Method

3.5.1 Fitness function

The fitness function f plays an important role in the genetic algorithm and it is used to evaluate the population and select the best population, it will be based on the function $f(S_{n+1})$, with S_{n+1} as the start time of the last activity i.e. the time to realize the project. The goal is to find the best result of the function, which is the short value of the start time of the last activity, saying that for each individual the result of f is checked. The mathematical formula is:

$$\mathbf{f_i} = 1/f(S_{n+1})$$

where $f(S_{n+1})$ is non-zero and i takes values from 1 to N for each individual in the population. In the case of our study, we take the negated duration of the project as the fitness function. Since we want to minimize the duration of the project, we take the less negative value as the higher fitness score. Individuals with higher score are with high probability qualified to be selected as candidates for further steps Methods of Selection.

3.5.2 selection method

The selection is an important task and critical operator used to determine which individuals from the current population are carried forward to the next generation. The first role of the selection process is to generate an improved population of solutions based on the existing ones. We have two most frequently used selection operators .
Introduced by Holland in 1975, it enables assigning selection probabilities to individuals based on their fitness score. Roulette wheel selection is a genetic algorithm operator used to select potentially useful solutions for recombination. The process begins by segmenting a circular wheel and designating a specific point on its circumference. The wheel is then set in motion, and the segment of the wheel before the designated point is selected as the first parent. This process is repeated to select the second parent as presented in the table (3.1) below [29][26].
Using this method, it can be seen that a fitter individual will have a larger slice of the wheel, which will increase the probability of landing near the fixed point

Individual	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Table 3.1: Individual and fitness value [26]

during the rotation.

The probability of selection of an individual is therefore directly influenced by its fitness. Higher fitness corresponds to a higher chance of selection.[20]
The steps of the roulette wheel selection is the following:[26] These steps can be

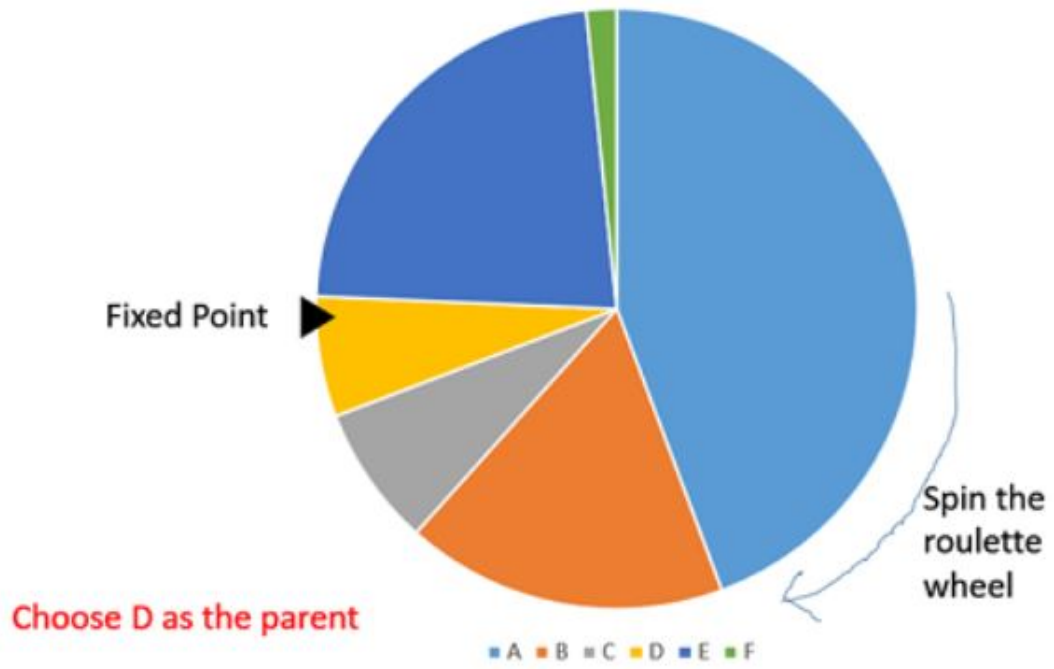

Figure 3.4: roulette wheel selection method for the table 2.1 [26]

specified as follows: [27][26]

- **step 1:** Calculate the fitness value for each chromosome.

- **step 2:** Find the total fitness of the population with:

$$\mathbf{f} = \sum_{j=1}^{n} F_j$$

, where F_j is the fitness value not equal to zero of each individual.

- **step 3:** Calculate the probability of selection P_i for each chromosome

$$\mathbf{P_i} = F_i / \sum_{j=1}^{n} F_j$$

, where F_i is the chromosome for which the probability of selection P_i is calculated [25].

- **step 4:** Calculate the cumulative probability C^i as the sum of P_j, where j ranges from 1 to i.

- **step 5:** Generate a random number R in the range [0-1].

- **step 6:**Apply the selection condition: if $R < C[1]$ then select chromosome [1], otherwise select chromosome [i] so that $C^{i-1} < R < C^i$

3.6 Crossover Operators

Crossover is a crucial genetic operator, that facilitates the fusion of two-parent entities to generate a new offspring. for the case of our investigation, we are replacing two parent chromosomes with child chromosomes. The idea behind crossover is to potentially produce offspring that outperform both parents by inheriting favorable traits from each other (Hartmann, 1998) [17]. In genetic algorithms, the crossover is typically applied with a high probability, called the crossover rate, denoted as α = 0.5. This value determines whether two parents cross over or whether the children remain unchanged. The crossover rate effectively determines the probability of performing crossover operations.

The selection of parents for crossover is randomized and influenced by a user-defined crossover probability or rate value. To meet the needs of permutation problems,in the context where every activity is numbered from 1 to n and should be represented only once in the produced offspring chromosomes, we investigate a range of genetic operators. The following crossover operators have been implemented during the case of our search.

1PX (One-Point Order Crossover): With this operator the selection of a random point within the length of the chromosome is done. As a result, both chromosomes are divided into two segments. The odd segment doesn´t change, while the even segment of each chromosome is rearranged according to the sequence of the other chromosome. This method is applied to a random key-based representation form with complete precedence relations.

To illustrate the one-point crossover, consider the case of an activity list and a random key. The process is further explained in the following example for random key-based representation form:[17]

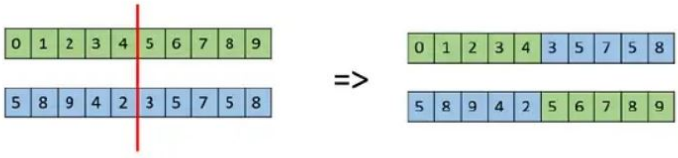

Figure 3.5: Overview One Point Crossover [9]

- **Using 1PX (One-Point Order Crossover) for the random key:** The one-point order crossover can also utilized in our case. We consistently select a random integer q such that $1 \leq q < J$. The initial position of the offspring individual D namely the daughter is inherited from the mother, while the father defines the remaining q. In other words, for each $i = 1, J$, we ultimately achieve: [17]

$$\begin{cases} pv_i^M, & \text{if } i \in \{1, \ldots, q\} \\ pv_i^F, & \text{if } i \in \{q+1, \ldots, J\}. \end{cases} \tag{3.1}$$

Let us take the example of the parents: the mother $M = (0.58, 0.64, 0.31, 0.87, 0.09, 0.34)$ and the father $F = (0.12, 0.43, 0.99, 0.65, 0.19, 0.22)$ with **one point crossover** in this case we get the daughter with $D = (0.58, 0.64, 0.31, 0.65, 0.19, 0.22)$ and for the son we get $S = (0.12, 0.43, 0.99, 0.87, 0.09, 0.34)$[17].

1PX (Uniform Crossover): This is the other type of crossover operator that we can apply in the case of the Resource Constraint Project Scheduling Problem. For each gene in both parents, we flip a coin to determine whether it will be included in the offspring or child[17].

Figure 3.6: Overview Uniform Crossover [9]

- **Application of uniform crossover for random key:** We generate a series of random numbers: $p_i \in \{0, 1\}$, $i = 1, \ldots, J$. For each $i = 1, \ldots, J$ we set:[17]

$$pv_i^D = \begin{cases} pv_i^M, & \text{if } p_i = 1, \\ pv_i^F, & \text{otherwise.} \end{cases} \tag{3.2}$$

let us take the same parents with mother M= (0.58, 0.64, 0.31, 0.87, 0.09,0.34) and the father F = (0.12, 0.43, 0.99, 0.65,0.19, 0.22) by applying the uniform crossover, which consists to each gene, to randomly choose whether to inherit from mother or father, building the children, we can have the random choices depending on the crossover rate for the selection of mother´s gene or father´s gene: (mother, father, mother, father, father, mother) and we can get as results: S= (0.58, 0.43, 0.31, 0.65, 0.19, 0.34).

PMX (Partially Mapped Crossover): It is another very important type of crossover operator that is used for activity list-based representation form by recombination and crossing genes from the parents' chromosomes to produce like other crossover operators offspring or children[16]. Partially mapped Crossover (PMX) is a genetic algorithm operator that involves exchanging genes or activities between two parents. The process begins by identifying two crossover points in the parent chromosomes. The genes or activities located between these points are then exchanged between the two parents. The remaining genes are mapped to their respective positions on the other parent chromosome, guided by the established mapping relationship determined by the exchanged genes. This process ensures that the relative order of the genes is preserved, which is a crucial aspect in permutation-based problems.PMX tries to maintain a precedence relationship of activity and enable the generation of a feasible schedule [40].

- **Application of Partially mapped crossover for activity list** Let's consider two parental chromosomes $M = (1, 3, 2, 5, 4, 6)$ and $F = (2, 4, 6, 1, 3, 5)$. To apply the Partially Mapped Crossover (PMX) operator, we first select a random region in both genomes, for example, from the second to the fourth element. Then, this region is swapped in both genomes, resulting in two new

genomes: (1,4,6,1,4,6) and (2,3,2,5,3,5). However, the same numbers occur multiple times in both genomes, which violates the permutation property. To address this issue, we perform another calculation step. We form all the pairs of numbers that have been swapped, i.e., 43, 62, and 15. We then swap these numbers with each other outside the range. For instance, we swap 4 with 3, 3 with 4, 6 with 2, 2 with 6, 1 with 5, and 5 with 1. Finally, we obtain two new genomes where each number appears only once.

child1 = (5,4,6,1,3,2)

child2 = (6,3,2,5,4,1).

We should emphasize that if the entire population selected for crossing comprises only one type of string, the crossover of two chromosomes does not produce any new ones[40].

3.7 Mutation Operators

Mutation is an important genetic operator for changing one or more gene values for a chromosome. Its use ensures that the population does not get stuck in local optima during the optimization process. The choice of chromosomes that are submitted to the mutation is random and determined by a user-defined mutation rate[20]. However, the traditional mutation operators may not be ideal for the Resource-Constrained Project Scheduling Problem due to the need to satisfy all precedence relationships within the project. To address this, modifications are made to the mutation operators to make them more suitable.
In the table 3.2, we present the Insertion mutation-operator implemented in the case of the activity-list-based representation form. The table 3.3 represents the swap and random reset mutation operators implemented in the case of random-key based representation form.

Application of Insertion Mutation Operator for activity list based representation form Given an initial genome I=[1,2,3,4,5] with a mutation rate of 0.3 if the random decision is less than 0.3, insert mutation is performed. The mutation operator randomly selects an element from the activity list, removes it, and then inserts it at a random position. In the example given:

, if the random decision is 0.2, i.e. less than 0.3, the insert mutation could occur, resulting in M= [1,3,2,4,5].

Mutation	Description	Example
Insertion	This mutation operator involves randomly selecting an activity and inserting it at a different position in the schedule. It allows us to reorder activities in the schedule	With a schedule [activity 1, activity 2, activity 3, activity 4, activity 5], we could randomly select activity 3 and insert it at activity 1, resulting in [activity 3, activity 1, activity 2, activity 4, activity 5].

Table 3.2: Description of Mutation Types

If the random decision is 0.6, that is greater than 0.3, the activity list will not change: with M=I= [1,2,3,4,5].

Table 3.3: Mutation Operators for RCPSP with Random key based representation form

Mutation	Standard Procedure	Modified Procedure for RCPSP
Swap	Exchange two randomly selected activities.	Step 1: Randomly select gene g (activities). Step 2: Find genes to the left and right of g to swap, respecting precedence relationships. Step 3: Swap gene g with a randomly selected gene from the set found in step 2 [20].
Random Reset	Assign randomly a value from the allowed set to a randomly selected gene or activity.	Step 1: Randomly select value g. Step 2: Identify random positions in the chromosome that satisfy precedence relationships. Step 3: Insert the randomly selected gene g into other genes.

Application of the random reset mutation operator to a random-key based representation form: Suppose we have an initial genome with three random values, I= [0.55, 0.75,0.32]. To apply the random resetting mutation operator, we set an initial mutation rate of 0.5, which means there is a 50 percent chance of mutation. We then make a random decision between 0 and 1. If the number of random decisions is less than the mutation rate of 0.5, we perform a

mutation by replacing a particular position in the genome with a new value. Otherwise, we keep the original value.

If the random decision is 0.3, which is less than 0.5, we decide to mutate the first value (0.55) by replacing it with a random value. The resulting genome after mutation is M = [0.21, 0.75, 0.32].

If the random decision is 0.8, i.e. greater than 0.8, the genome remains the same and we obtain M = I = [0.55, 0.75, 0.32]. Note that after the mutation, the resulting genome is encoded for the next operation.

Application of the random swap mutation operator to a random-key based representation form: Suppose the initial genome I = [0.55. 0.75, 0.32, 0.91], the mutation rate is 50 percent or 0.5, we have for example these two cases: If the random decision is 0.3 and therefore less than the mutation rate of 0.5, two random elements could be swapped, giving the result M = [0.32,0.75,0.55,0.91]. The values 0.32 and 0.55 have been swapped .
If the random decision is 0.6, the genome does not change and we get M = I = [0.55, 0.75, 0.32, 0.91].

Chapter 4

Experimental Setup and Application of Genetic Algorithm

This phase aims to apply the genetic algorithms to the solution of an instance problem, based on the two representation forms described: the activity list-based representation form and the random key-based representation form. Each of the algorithms will be applied to the defined instance problem, facilitating, in practical steps, the understanding of each form of genetic algorithm defined in the previous chapters. The steps for solving the instance problem using genetic algorithms will be as follows

- **Define the RCPSP Problem:** This step involves specifying project tasks, their duration, and priority, as well as defining available resources and their capacities.

- **Definition of the initial population:** To generate the initial population for the genetic algorithm, this step starts by considering the population size with three individuals as a parameter set. These individuals are represented as a chromosome using one of each representation form between activity list and random key.

- **Genetic Operators:** This step involves the following five sub-steps: **Initialization**, **Selection**, **Crossover**, **Mutation**, and **Replacement**.

  Firstly, the **Initialization** sub-step sets up the initial population of plans.

  Secondly, the **Selection** sub-step selects parent schedules based on their fitness for reproduction.

  Thirdly, the **Crossover** sub-step combines information from two parent plans to produce offspring or a child.

Fourthly, the **Mutation** sub-step introduces small random changes to some plans in the population.

Finally, the **Replacement** sub-step replaces less fit individuals with newly generated offspring

- **Fitness Function:** A fitness function is defined to evaluate how well a schedule meets the project objectives and constraints. It should take into account factors such as project duration, resource utilization, and violation of constraints.[43]

- **Termination Criteria:** We define stopping criteria when we decide about the maximum number of generations or when we reach a satisfactory solution.

4.1 Cases Study

We are considering a PSPLIB instance in the form of an MPM (Metra Potential Method) network plan (see Figure 3.1) with four real activities and two dummy activities (0 and 5). The plan also includes two renewable resources, r_1 and r_2, with corresponding resource capacities of $R_1 = 2$ and $R_2 = 3$ and the maximum project duration is 8.

Based on the representation of the instance problem, the important data required to solve it can be presented in the following table4.1:

Table 4.1: instance problem

Job i	Processing Time	Number of Successors	Successor j	r_{i1}	r_{i2}
0	0	2	1, 3	0	0
1	1	1	2	2	2
2	1	2	4, 5	2	2
3	4	1	2	1	2
4	2	1	5	1	1
5	0	-	-	0	0

To be solved by genetic algorithms, an instance needs to be readable using the **readInstancePSPLIB.py** module, for the project scheduling problem library is generally presented in the following form. Note that in the instance data, all activities are numbered from 1 to 6, with 1 and 6 being dummy or fictitious activities.

```
file with basedata            : j30_29.bas
initial value random generator: 1958
projects                      :  1
jobs (incl. supersource/sink ):  6
horizon                       :  8
RESOURCES
  - renewable                 :  1   R
  - nonrenewable              :  0   N
  - doubly constrained        :  0   D
PROJECT INFORMATION:
pronr.  #jobs rel.date duedate tardcost  MPM-Time
    1      4      0       8       14       8
PRECEDENCE RELATIONS:
jobnr.    #modes  #successors   successors
   1        1          2           2   4
   2        1          1           3
   3        1          2           5   6
   4        1          1           3
   5        1          1           6
   6        1          0
REQUESTS/DURATIONS:
jobnr. mode duration  R 1  R 2

  1      1     0       0    0
  2      1     1       2    2
  3      1     1       2    2
  4      1     4       1    2
  5      1     2       1    1
  6      1     0       0    0

RESOURCE AVAILABILITIES:
  R 1  R 2
```

4.2 Time permissible Schedule

To find and explain the schedule of the given example, we will use the serial generation scheme. Firstly, we find the **earliest start (ES)** and **latest start (LS)** time, which are essential for the application of the serial generation scheme. The results of this process are shown below for the given instance example using the triple Algorithm:

i	0		1		2		3		4		5	
	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}	d_{ij}	p_{ij}
0	0	0	0	0	4	3	0	0	5	2	7	4
1	-4	5	0	1	1	1	-4	0	2	2	4	4
2	-5	5	-5	0	0	2	-5	0	1	2	3	4
3	-1	5	-1	0	4	3	0	3	5	2	7	4
4	-6	5	-6	0	-2	3	-6	0	0	4	2	4
5	-8	5	-8	0	-4	3	-8	0	-3	2	0	5

Figure 4.1: results of triple Algorithm

Based on the results of the triple algorithm in table 4.1, we get the derived table 4.2 containing the earliest and latest start times of each activity.

Table 4.2: Earliest and Latest Starting Time of Activities

$i \in V$	0	1	2	3	4	5
ES_i	0	0	4	0	5	7
ES_i	0	4	5	1	6	8

Secondly, we obtain a feasible schedule of the project instance with activities: $V = [0, 1, 2, 3, 4, 5]$, resource requirements:[0,2,2,1,1,0] and duration [0,1,1,4,2,0] by applying the serial schedule generation scheme described step by step from Figure4.2,4.3 to Figure 4.4 as follows:

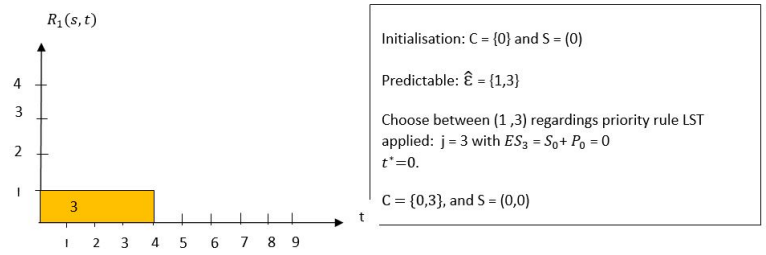

Figure 4.2: Initialisation Serial Schedule Generation Scheme

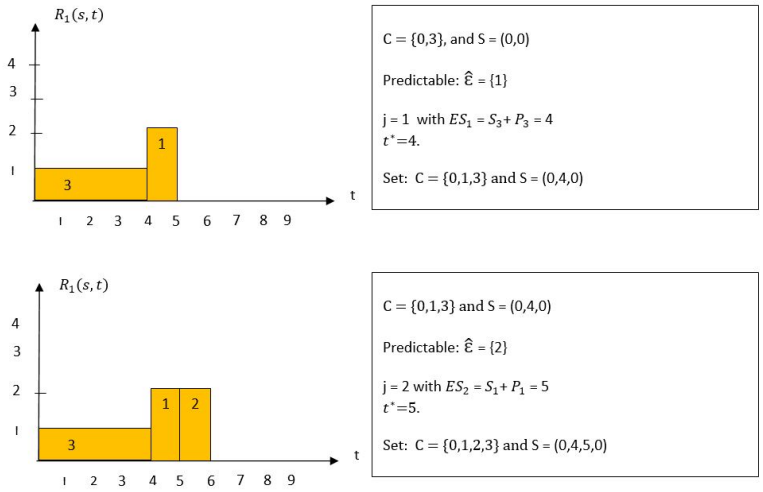

Figure 4.3: Steps Serial Schedule Generation Scheme

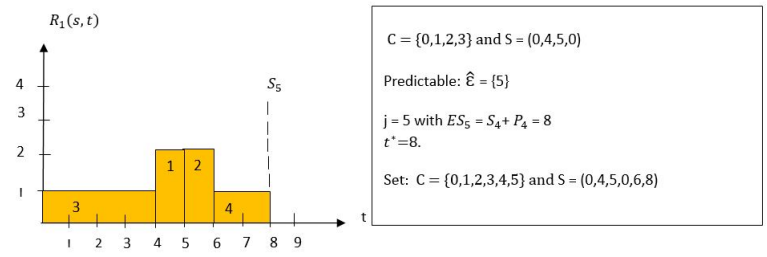

Figure 4.4: Result Serial Schedule Generation Scheme

4.3 Application of Genetic Algorithm: Definition of the initial population

With a population size of three, we first generate three individuals to be used for the next stage of fitness evaluation in the genetic algorithm. This process begins after reading the problem instance with the readInstancePSPLIB.py code. After this first instance reading step and using the Genome.py module, the next three individual vectors are generated using the activity list representation, which shows a prioritized list of tasks, with the number of each individual indicating the order in which the tasks should be performed.

- **For the given instance project example an activity list representation** are initially generated with three individuals, these activities follow schedule:

$I_1 =$ **[1 3 2 4 5]**
$I_2 =$ **[1 3 2 4 5]**
$I_3 =$ **[3 1 2 4 5]**

- **For the given instance project a random key representation** followed by different individuals can be generated to go for [0,1] values:
  $I_1 =$ **[0.8196961029958932, 0.5665694275828908, 0.21399464535039459, 0.18773661364395589, 0.41392904951095666]**
  $I_2 =$ **[0.0017205867549360265, 0.9830414865063575, 0.3798066253905522, 0.36203119626809144, 0.3850945483423729]**
  $I_3 =$ **[0.11810379243740621, 0.02717236491653885, 0.6470132342659467, 0.5315926972308601, 0.673924548692638]**

4.3.1 Evaluation of best individuals using fitness function

The fitness function is used to select the best individuals for the crossover, allowing us to accurately evaluate the best completion times for the individuals initially generated in the previous section. In our case, the selection method is the roulette wheel. As part of the genetic algorithm code, we evaluated the fitness function of the three individuals or genomes. This function called the Negative Time Fitness Function, gives three negative values to select two of the most negative results for the subsequent crossing and mutation stages. The application of the roulette wheel Selection gives the following results: [-8, -8, -8] The same result is obtained in the case of this example for the second, third, and subsequent generations.

4.3.2 Selection of best individuals

After evaluating each genome and its fitness, we select the two best individuals using the selection function called roulette wheel selection (fitnessscores), which in this case selects the two best fitness values. The two best individuals are Father $I1$ = [1 3 2 4 5] and Mother $I2$ = [1 3 2 4 5] with the best negative duration as a fitness score of -8 1. As noted, [1 3 2 4 5] represents the genome with the optimal result and parents with the same activity list order can be selected for crossing and mutation.

4.3.3 Crossing of best individuals

After selecting two best individuals, these are subjected to crossover using the Partially-Mapped-Crossover operator for activity list representation or the Uniform Crossover or One-Point Crossover for random key representation based form

following children obtained from parents (father and mother) can be crossed using the Partially-Mapped-Crossover operator: [1 3 2 4 5] [1 3 2 4 5] are children or offspring obtained from parents [1 3 2 4 5] [1 3 2 4 5].

4.3.4 Mutation of best individuals

After we find the best individuals as children, each of them is mutated with the corresponding insert mutation operator used for the activity list in this case.The result of the children after mutation has not changed because the optimal result has been reached and the probability of changing an activity place is very low because the priorities for executing activities are respected in this case and cannot be changed.We get for child1 = [3 1 2 4 5] and for child2 = [3 1 2 4 5].

4.3.5 Generation of Best-time And Best Schedule And Next Iteration

The generation of the best result is based on the mutated children and provides the best schedule and the best scheduled time, which is the duration of the last job of the project instance and represents the most minimized duration of the project instance. We can represent this result as a dictionary as follows: [0: 0, 3: 0, 1: 4, 2: 5, 4: 6, 5: 8], where each key has an associated duration. The generated schedule can also be represented as a list: (0, 0, 4, 5, 6, 8) representing the duration of implementation of each activity represented as follows (0, 3, 1, 2, 4, 5), with the best duration being 8 for the realization of the project instance.The following plot after 100 generations with parameters such as number of generations = 1, size of generation = 50 and mutation rate = 0.05 is generated as shown in figure 4.5:

Once the results have been obtained, the algorithm moves on to the next iteration according to the number of generations defined. To achieve the optimal solution, it is essential to impart this knowledge specifically to the succeeding generations.The results obtained in our case in the first iteration remain the same after several generations, because optimality has already been reached, and remain at 8, that is why the line remains at time 8 after 100 generations, as shown in the graph.

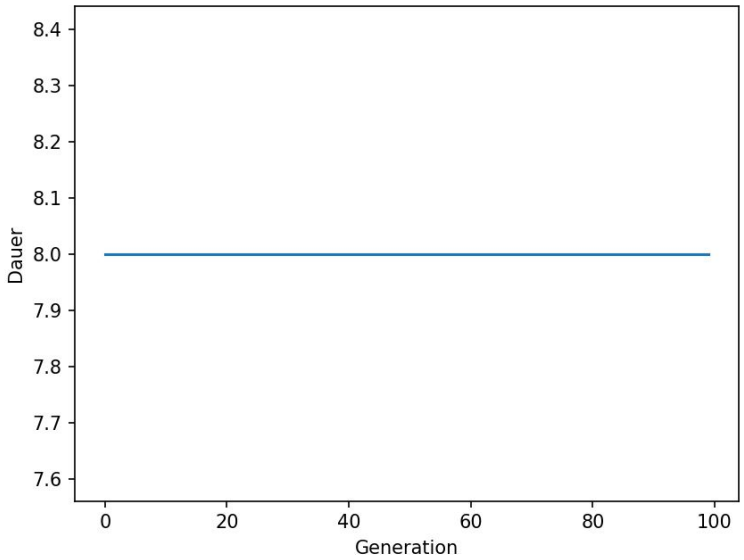

Figure 4.5: Plot of Genetic Algorithm Solution after 100 Generations of Instance Example

Chapter 5

Results and Analysis

This chapter will concentrate on the results of the algorithm portfolio approach implemented to address the challenges posed by the Resource-Constrained Project Scheduling Problem. In the previous chapters, we examined the theoretical foundations of the problem of scheduling a project under resource constraints, by detailing the design and implementation of the algorithm portfolio, and superficially describing the experimental setup. We now present the results obtained by running the algorithm on a set of designed test instances.

5.1 Programming Environment for Genetic Algorithms in the Resource Constrained Project Scheduling Problem Algorithm Portfolio Approach

We will focus on the programming environment established for the implementation of genetic algorithms as a key component of the algorithm portfolio solving Resource-Constrained Project Scheduling Problem. The programming environment will play an important role in the successful development, testing, and evaluation of our algorithm portfolio. It will provide an overview of the chosen programming language, libraries, tools and best practices used to create the programming environment

5.1.1 Programming Language Selection

The selection of a programming language is a crucial aspect of the development process. We opted for Python, a programming language for several reasons. It

is renowned for its user-friendly syntax and readability. The choice of programming language is necessary for the development process. We chose Python, the programming language for reaching the goal because of several reasons, Python is known because of its **ease of use** and **readability**. With its gentle learning curve, Python proves advantageous in academic settings where interdisciplinary collaboration is prevalent. The language facilitates the logical integration of genetic algorithm code with various libraries and tools for data analysis, visualization, and reporting. This capability is important for experiments and to present results with high effectivity. Python´s dynamic typing and its interpreted nature make it an excellent choice for rapid prototyping and experimentation with diverse algorithms and parameters. Additionally, Python includes easy interfaces including codes from other languages like C/C++, allowing for flexibility without compromising overall convenience.[12]

5.2 Execution Environment

With the goal of simplifying the development process and ensuring efficient code management, we have chosen to use an Integrated Development Environment (IDE), specifically Visual Studio Code version 1.81.0 as the Integrated Development Environment. The running of the algorithm can be implemented based on the following aspects:

- **operating system** mostly Windows 10 or 11, or Linux Ubuntu.
- **capacities** Usually a multi-core processor (12 CPU cores), sufficient memory (Random Access Memory RAM 16, 32 or 64 gigabytes, hard drive 1 terabyte), and a powerful graphics card.

5.2.1 Genetic Algorithm Libraries

The main libraries for implementing the genetic algorithm are:

- **random module** : The seed determines the sequence of numbers that are generated, so they are pseudo-random. If the seed does not change, neither does the sequence.
- **Multiprocessing:** work with processes for achieving parallelism. It allows the program to execute tasks concurrently, taking advantage of multiple processes to speed up the overall execution.
- **time:**it counts the time of execution of each instance and enables also to get the overall processing time if needed.

5.2.2 Problem-Specific Libraries

for the studied case, certain libraries or modules may be developed to handle problem-specific tasks, including:

- Resource Constrained Project Scheduling Problem: Data Representation: Developing data structures and classes to effectively represent Resource Constrained Project Scheduling Problem instances. It is the case of **readInstancePSPLIB** which implements the lecture of instance and the storage of value in different classes: resource, mode, job and problem.

- Fitness Function Evaluation: For the implementation of functions to evaluate the fitness of RCPSP schedules based on the constraints and objectives of the problem.

5.2.3 Visualization and Analysis

- matplotlib.pyplot and seaborn: is a collection of functions that make matplotlib work like MATLAB.Each Pyplot function makes some change to a plot: for example, creates a plot, creates a plot area in a plot, plots some lines in a plot area, decorates the plot with labels, generates graphs or diagrams, and presents results. It is used in the case of our search for a graph showing how the best solution found has changed over time.

5.3 Parameter Tuning and Experimentation

To set the genetic algorithm and evaluate its performance, we carried out rigorous experiments, and we adjust various parameters, including:

1. **Population size:** Choosing the population size is very important for Genetic Algorithms. It can influence how well and how fast the algorithm solves the Resource-Constrained Project Scheduling Problem. The population size tells us how many solutions (also called individuals or chromosomes) are tested and improved in each genetic algorithm cycle.

2. **Crossover rate:** The crossover rate is an important factor in genetic algorithms. It affects how individuals (chromosomes) share genetic information during evolution. For our research on solving the Resource-Constrained Project Scheduling Problem using Genetic Algorithms, the crossover rate will influence many aspects such as exploration and exploitation, population diversity, good solution retention, solution quality, algorithm speed, parameter interaction, and testing.

3. **Mutation rate:** The mutation rate is key for genetic algorithms. It helps create genetic diversity in the population as it evolves. For optimization problems with genetic algorithms, the mutation rate affects many points, like exploring and diversifying, avoiding early convergence, keeping genetic diversity, improving solutions, combining crossover, interacting parameters, speeding up convergence, testing and tuning empirically, and adapting to problem features.

4. **termination criteria in Genetic Algorithms:** conditions or rules that decide when the algorithm should stop its search and return the best solutions found. In the context of the search to solve the Resource-Constrained Project Scheduling Problem using Genetic Algorithms. For example, we set a maximum number of generations, i.e. terminating the genetic algorithm after a certain number of generations using the parameter, several generations (e.g. 100 generations), this sets an upper limit on the number of iterations the genetic algorithm can perform. This is useful for controlling computational resources and ensuring that the algorithm does not run indefinitely. The roles of termination criteria in the case of genetic algorithms are:

   (a) **Controlling resources:** The termination criteria save computing resources by stopping the algorithm from running too long or using too much computing time.

   (b) **Preventing Overfitting:** They avoid fitting the training data too closely or optimizing too much, and make the algorithm work well on new instances. Termination criteria affect how much the algorithm explores and exploits. For instance, a convergence criterion may tell when to stop.

   (c) **Balancing Exploration and Exploitation:** Termination criteria control how the algorithm searches and uses information. For example, a convergence criterion may indicate when to end.

   (d) **Adapting to Problem Characteristics:** The algorithm can perform better by using termination criteria that suit the RCPSP instances' features.

5.4 Resources Constrained Project Scheduling Problem instances for the algorithm portfolio

To evaluate the portfolio of algorithms, different classes of instances can be used that require resolution and evaluation: the algorithm is tested on sets

of instances from the J120 case study, consisting of activities systematically generated by the ProGen project generator developed by Kolisch et al.[23]. This class was chosen because of the large number of instances and activities it contains and the large space of results it can provide, allowing its results to be analyzed and exploited. There are different case studies involved by PSPLIB in sm format with 480 instances for categories J30 and J60 and for case study 90, 600 problem instances were involved.[34] The numerical experiment is carried out on the instance sets for case study J120 by performing 100 and 500 solution evaluations after one run for each instance.

5.5 Automatisation of Solutions and Analysis

The experimental design will include a variation of configurations defined to build the genetic algorithm portfolio to execute instances for case study j120. These configurations represent different algorithms for building the portfolio based on the approach of the Genetic Algorithm and the parameter all suitable for solving instance problems fo RCPSP. This part will be based on the exploration of the results of sets of instances for the Project Scheduling Problem Library (PSPLIB) for the analysis of each of the two genetic algorithms-based representation forms. First, we will explain the two main modules created, **configuration. py** and **simulation. py**. The simulation module contains different parameters to define configurations to run each given project instance and compare the result; however, different settings such as genome type, number of generations, generation size, mutation operator, mutation rate, crossover operator, and crossover rate [2] should be defined to run a different list of project instances for study case j120. The second module **configuration. py**, contains the operators and parameters needed to solve a problem; this parameter can be improved in the computation before and after execution. The most important operators were the mutation operator with random reset, random swap and insert, and the crossover operator: Uniform, One point, and Partially mapped, all used for the two representation forms: activity list and random key. The implementation was based on all instances included in the study case j120. First, we defined and modified the values to create each different configuration to solve these instances. First, we present four different configurations or algorithms using the two representation forms in computation.py:

The table 5.1 summarizes all required configurations building the portfolio and they will be used to solve problem instance projects contained in case

study j120:

Table 5.1: Configurations

Configurations	Genomtyp	Nber Gen.	Gen. size	Mut. operator	Mut. rate	Cros. Operator	Cros. rate
Config. 1	Random-Vector	50	50	Random-Reset.	0.05	Uniform	0.7
Config. 2	Random-Vector	50	50	Random-Reset.	0.05	One Point	0.7
Config. 3	Activity-list	50	50	Random-swap	0.05	Partially-mapped	0.7
Config. 4	Activity-list	50	50	Insert	0.05	Partially-Mapped	0.7

5.6 Results Analysis

The following analysis presents the results obtained from the implementation of all configurations defined in Table 5.1, including the two representation forms, solving 497 instances included in case study j120. This analysis is based on two important performance metrics:

- **Makespan:** It is one of the performance measures for the RCPSP. representing the total time required to complete all projects. A lower makespan indicates better performance and we look for the optimal solution, i.e. the lowest makespan.

- **computation time:** This metric measures the efficient time of execution of each instance during the planning process.

5.6.1 results of instances for case study j120:

From the 497 results generated, we took 100 instances as part of our analysis of the study case J120. for the four configurations using the random key and activity list representation forms, these results are presented in Tables 5.2 and 5.3 below:

In this results following observations , comparison and analysis can emerge from it: Through these results, we observe a variation of the *best time* for each instance executed by each of the four configurations defined. For example, if we take the case of the first instance j12011-6.sm, we observe a better result of *248* as the best optimization time provided by configuration 4, which includes the activity list representation form. We then observe a variation of better results between configurations 3 and 4 for the other instances, which use the activity list-based representation form and offer

Table 5.2: results of 100 instances of case study j120

Instance	Best time Config. 1	Comp. Time	Best time config. 2	Comp. Time	Best time Config. 3	Comp.Time	Best time Config. 4	Com
InstanzenPSPLIB/j12011_6.sm	263	2.631 min	269	2.611 min	250	2.742 min	248	2.7
InstanzenPSPLIB/j12011_2.sm	192	2.658 min	193	2.667 min	185	2.741 min	188	2.7
InstanzenPSPLIB/j12011_1.sm	209	2.647 min	207	2.665 min	200	2.792 min	202	2.7
InstanzenPSPLIB/j12011_7.sm	201	2.657 min	202	2.658 min	194	2.817 min	194	2.7
InstanzenPSPLIB/j12011_9.sm	212	2.701 min	206	2.684 min	205	2.792 min	200	2.7
InstanzenPSPLIB/j12011_8.sm	199	2.647 min	201	2.692 min	189	2.805 min	191	2.8
InstanzenPSPLIB/j12011_4.sm	246	2.686 min	245	2.684 min	232	2.804 min	232	2.8
InstanzenPSPLIB/j12011_5.sm	266	2.670 min	265	2.700 min	250	2.810 min	250	2.8
InstanzenPSPLIB/j12011_10.sm	220	2.669 min	226	2.696 min	214	2.827 min	215	2.8
InstanzenPSPLIB/j12011_3.sm	254	2.738 min	252	2.744 min	242	2.854 min	242	2.8
InstanzenPSPLIB/j12012_2.sm	137	2.559 min	139	2.634 min	133	2.667 min	136	2.6
InstanzenPSPLIB/j12012_4.sm	153	2.576 min	152	2.648 min	145	2.685 min	145	2.7
InstanzenPSPLIB/j12012_6.sm	149	2.572 min	146	2.639 min	141	2.744 min	142	2.7
InstanzenPSPLIB/j12012_3.sm	160	2.650 min	162	2.678 min	154	2.745 min	156	2.7
InstanzenPSPLIB/j12012_8.sm	143	2.598 min	143	2.733 min	139	2.759 min	139	2.7
InstanzenPSPLIB/j12012_9.sm	129	2.645 min	129	2.717 min	122	2.822 min	122	2.7
InstanzenPSPLIB/j12012_7.sm	143	2.673 min	143	2.767 min	139	2.742 min	140	2.8
InstanzenPSPLIB/j12012_1.sm	166	2.695 min	166	2.836 min	161	2.830 min	161	2.8
InstanzenPSPLIB/j12012_10.sm	167	2.634 min	166	2.719 min	161	2.699 min	159	2.7
InstanzenPSPLIB/j12012_5.sm	199	2.674 min	201	2.780 min	195	2.851 min	195	2.8
InstanzenPSPLIB/j12013_1.sm	150	2.649 min	153	2.678 min	143	2.758 min	148	2.7
InstanzenPSPLIB/j12013_2.sm	102	2.629 min	102	2.622 min	99	2.723 min	98	2.7
InstanzenPSPLIB/j12013_5.sm	110	2.552 min	108	2.541 min	104	2.687 min	104	2.6
InstanzenPSPLIB/j12013_4.sm	133	2.576 min	133	2.616 min	126	2.785 min	126	2.7
InstanzenPSPLIB/j12013_3.sm	143	2.674 min	141	2.661 min	139	2.769 min	139	2.8
InstanzenPSPLIB/j12013_6.sm	120	2.629 min	121	2.591 min	113	2.698 min	114	2.6
InstanzenPSPLIB/j12013_8.sm	113	2.679 min	114	2.588 min	113	2.750 min	111	2.6
InstanzenPSPLIB/j12013_9.sm	102	2.704 min	103	2.656 min	98	2.828 min	99	2.8
InstanzenPSPLIB/j12013_10.sm	114	2.651 min	114	2.719 min	109	2.830 min	109	2.7
InstanzenPSPLIB/j12013_7.sm	131	2.720 min	129	2.702 min	124	2.835 min	125	2.8
InstanzenPSPLIB/j12014_2.sm	111	2.487 min	112	2.532 min	107	2.690 min	106	2.6
InstanzenPSPLIB/j12014_3.sm	99	2.559 min	98	2.591 min	96	2.789 min	96	2.6
InstanzenPSPLIB/j12014_4.sm	104	2.590 min	104	2.560 min	103	2.731 min	102	2.6
InstanzenPSPLIB/j12014_1.sm	101	2.616 min	100	2.697 min	98	2.822 min	101	2.7
InstanzenPSPLIB/j12014_7.sm	106	2.550 min	105	2.568 min	102	2.710 min	102	2.6
InstanzenPSPLIB/j12014_6.sm	103	2.632 min	104	2.564 min	100	2.804 min	99	2.7
InstanzenPSPLIB/j12014_5.sm	118	2.684 min	119	2.689 min	114	2.812 min	116	2.8
InstanzenPSPLIB/j12014_9.sm	109	2.608 min	110	2.608 min	104	2.781 min	105	2.7
InstanzenPSPLIB/j12014_8.sm	130	2.654 min	128	2.631 min	125	2.755 min	126	2.7
InstanzenPSPLIB/j12014_10.sm	93	2.623 min	94	2.635 min	90	2.865 min	89	2.8
InstanzenPSPLIB/j12015_1.sm	84	2.557 min	87	2.534 min	81	2.622 min	81	2.6
InstanzenPSPLIB/j12015_2.sm	84	2.595 min	83	2.566 min	83	2.625 min	83	2.6
InstanzenPSPLIB/j12015_4.sm	90	2.617 min	89	2.536 min	87	2.676 min	90	2.6
InstanzenPSPLIB/j12015_3.sm	97	2.650 min	99	2.579 min	95	2.700 min	94	2.7
InstanzenPSPLIB/j12015_5.sm	87	2.612 min	87	2.579 min	87	2.710 min	87	2.7
InstanzenPSPLIB/j12015_6.sm	97	2.739 min	97	2.677 min	97	2.813 min	97	2.7
InstanzenPSPLIB/j12015_7.sm	75	2.695 min	77	2.622 min	75	2.767 min	75	2.7
InstanzenPSPLIB/j12015_8.sm	126	2.614 min	126	2.620 min	126	2.702 min	126	2.6
InstanzenPSPLIB/j12015_10.sm	100	2.578 min	96	2.614 min	97	2.704 min	97	2.7
InstanzenPSPLIB/j12015_9.sm	109	2.668 min	109	2.694 min	109	2.768 min	109	2.7
InstanzenPSPLIB/j12016_2.sm	281	2.610 min	280	2.596 min	272	2.735 min	274	2.7
InstanzenPSPLIB/j12016_1.sm	236	2.737 min	239	2.753 min	229	2.850 min	231	2.8
InstanzenPSPLIB/j12016_3.sm	285	2.638 min	286	2.637 min	275	2.808 min	274	2.8
InstanzenPSPLIB/j12016_4.sm	237	2.593 min	241	2.604 min	231	2.838 min	228	2.8

Table 5.3: results of 100 instances of case study j120

Instance	Best time Config.1	Comp. Time	Best time config.2	Comp. Time	Best time Config.3	Comp.Time	Best time Config.4	Comp. T
InstanzenPSPLIB/j12019_3.sm	98	2.513 min	96	2.524 min	94	2.645 min	94	2.621
InstanzenPSPLIB/j12019_2.sm	97	2.643 min	100	2.657 min	94	2.768 min	94	2.797
InstanzenPSPLIB/j12019_4.sm	122	2.605 min	124	2.580 min	123	2.695 min	121	2.692
InstanzenPSPLIB/j12019_5.sm	121	2.671 min	123	2.730 min	118	2.842 min	119	2.850
InstanzenPSPLIB/j12019_6.sm	107	2.641 min	108	2.615 min	102	2.741 min	100	2.813
InstanzenPSPLIB/j12019_7.sm	105	2.723 min	106	2.640 min	102	2.781 min	105	2.838
InstanzenPSPLIB/j12019_10.sm	97	2.639 min	98	2.665 min	93	2.800 min	94	2.725
InstanzenPSPLIB/j12019_8.sm	110	2.670 min	112	2.712 min	106	2.805 min	103	2.792
InstanzenPSPLIB/j12019_9.sm	101	2.720 min	103	2.774 min	99	2.873 min	97	2.899
InstanzenPSPLIB/j12020_1.sm	104	2.569 min	107	2.628 min	102	2.749 min	102	2.684
InstanzenPSPLIB/j12020_2.sm	104	2.598 min	103	2.643 min	101	2.708 min	103	2.685
InstanzenPSPLIB/j12020_3.sm	88	2.559 min	89	2.634 min	88	2.658 min	87	2.698
InstanzenPSPLIB/j12020_4.sm	91	2.614 min	90	2.645 min	89	2.680 min	89	2.708
InstanzenPSPLIB/j12020_5.sm	75	2.564 min	76	2.660 min	75	2.698 min	75	2.706
InstanzenPSPLIB/j12020_6.sm	80	2.694 min	80	2.660 min	80	2.794 min	80	2.790
InstanzenPSPLIB/j12020_7.sm	81	2.764 min	81	2.674 min	81	2.777 min	81	2.780
InstanzenPSPLIB/j12020_9.sm	81	2.597 min	80	2.611 min	83	2.663 min	83	2.669
InstanzenPSPLIB/j12020_8.sm	122	2.746 min	122	2.647 min	119	2.821 min	118	2.796
InstanzenPSPLIB/j12020_10.sm	88	2.738 min	88	2.644 min	83	2.835 min	84	2.804
InstanzenPSPLIB/j12018_6.sm	153	2.771 min	156	2.792 min	151	2.892 min	150	2.911
InstanzenPSPLIB/j12018_7.sm	138	2.605 min	138	2.624 min	134	2.744 min	132	2.739
InstanzenPSPLIB/j12018_10.sm	116	2.647 min	119	2.625 min	113	2.729 min	112	2.809
InstanzenPSPLIB/j12018_9.sm	105	2.634 min	106	2.644 min	102	2.831 min	102	2.750
InstanzenPSPLIB/j12018_8.sm	125	2.734 min	125	2.752 min	119	2.802 min	120	2.877
InstanzenPSPLIB/j12019_1.sm	99	2.656 min	100	2.590 min	96	2.729 min	97	2.668
InstanzenPSPLIB/j12017_4.sm	143	2.593 min	142	2.588 min	136	2.732 min	140	2.690
InstanzenPSPLIB/j12017_5.sm	156	2.647 min	154	2.682 min	150	2.789 min	153	2.839
InstanzenPSPLIB/j12017_6.sm	158	2.543 min	158	2.568 min	150	2.689 min	153	2.667
InstanzenPSPLIB/j12017_8.sm	148	2.697 min	150	2.601 min	143	2.768 min	143	2.755
InstanzenPSPLIB/j12017_7.sm	175	2.733 min	174	2.776 min	166	2.870 min	165	2.878
InstanzenPSPLIB/j12017_10.sm	156	2.709 min	153	2.680 min	152	2.799 min	153	2.827
InstanzenPSPLIB/j12017_9.sm	160	2.688 min	163	2.702 min	156	2.853 min	157	2.839
InstanzenPSPLIB/j12018_1.sm	157	2.594 min	156	2.584 min	150	2.698 min	150	2.727
InstanzenPSPLIB/j12018_2.sm	135	2.596 min	139	2.659 min	137	2.765 min	138	2.710
InstanzenPSPLIB/j12018_4.sm	116	2.650 min	118	2.581 min	114	2.685 min	114	2.722
InstanzenPSPLIB/j12018_3.sm	113	2.664 min	114	2.587 min	110	2.763 min	111	2.781
InstanzenPSPLIB/j12018_5.sm	136	2.522 min	138	2.593 min	133	2.696 min	133	2.740
InstanzenPSPLIB/j12016_5.sm	242	2.670 min	242	2.596 min	233	2.745 min	238	2.793
InstanzenPSPLIB/j12016_7.sm	221	2.679 min	224	2.625 min	215	2.799 min	214	2.805
InstanzenPSPLIB/j12016_6.sm	247	2.697 min	249	2.704 min	234	2.900 min	236	2.882
InstanzenPSPLIB/j12016_9.sm	249	2.625 min	247	2.655 min	238	2.853 min	237	2.800
InstanzenPSPLIB/j12016_8.sm	236	2.749 min	236	2.805 min	230	2.922 min	227	2.925
InstanzenPSPLIB/j12016_10.sm	254	2.705 min	258	2.699 min	247	2.898 min	249	2.857
InstanzenPSPLIB/j12017_1.sm	166	2.633 min	170	2.589 min	160	2.774 min	160	2.751
InstanzenPSPLIB/j12017_2.sm	144	2.613 min	145	2.633 min	140	2.742 min	139	2.782
InstanzenPSPLIB/j12017_3.sm	125	2.600 min	128	2.587 min	121	2.737 min	122	2.729

a better makespan than configurations 1 and 2 using the random key-based representation form. This could explain how the mutation or crossover rate of the best individuals can affect the individuals' representations. After varying the mutation or crossover rate, or increasing the number of individuals and generations, configurations based on the activity list still give better results than their predecessors. Regarding the execution time of the instances, the random key representation form shows a quick execution of a given instance, while the activity list representation form seems to be longer despite the better results offered.

5.6.2 Conclusion

In general all Configurations including the two representation forms offered a variety of results for case study j120. However there are sometimes one or two Configurations with a specified representation form mostly Activity List, where an instance can provide a best time or best duration, an overview can be given regarding parameter considerations, as follows:
Operator Rate:comparing both Configurations between Configuration 4 and 3, both using activity list based representation form, focus on high efficiency, may take advantage of a reduced operator rate.
Mutation Rate: Here's a possible reformulation of the text: "Configuration 2, which uses a random key-based representation, benefits from a moderate mutation rate to promote diversity. On the other hand, configuration 4 may benefit from a slightly lower mutation rate to refine solutions more efficiently.

Chapter 6

Algorithm Selection

The main point in this part will be based on the concept of automatic algorithm selection for a given instance of the problem. We have implemented two genetic algorithms based on two forms of representation, using four configurations that take them into account to determine which can be selected to solve a given instance problem. Configuration is thus defined on the basis of representations between a random key and a list of activities.

Second, we describe how the algorithm-based representation form included in each configuration is selected using one of the two genetic algorithms implemented for the portfolio with the four configurations created to solve the case study j120 using a machine learning approach. This allows the selection of the best configuration for a given instance problem. The machine learning approach is introduced for this important part where five models are presented and evaluated using the defined configuration and evaluation metrics of each model to achieve the desired goal. At the end of this phase, we will be able to select the most important model and train it to optimize our results. Next, we will describe the implementation framework and tools used, and give details of the selected model and its parameters. To understand the value of the portfolio framework, we will examine the performance measures and finally compare them with the results of the trained model.

6.1 Algorithm Selection and Machine Learning-Based Optimization

Combined Optimisation Problems (COPs) face the challenge of selecting the most appropriate algorithm to efficiently solve the given problem.[19] Algorithm Selection Problems (ASPs), a subset of machine learning, contribute to this challenge by automating the algorithm selection process based on historical data and problem characteristics. The research of Hutter et al.(2006)[18] has provided an opening for ASPs by establishing a framework that integrates the problem space and involves the extraction of metadata and the creation of a metamodel through meta-learning, thus enabling prediction of the optimal algorithms for specific problem instances.

6.1.1 Portfolio of Genetic Algorithms

As part of our research, we present a portfolio of algorithms, in particular two genetic algorithms (GAs) based on different activity representations: random keys and activity lists. The aim of this portfolio is to improve the adaptability and robustness of algorithmic solutions to the different instance problems of the case study j120.

6.1.2 Supervised Learning for Algorithm Selection

Machine learning as part of artificial intelligence uses algorithmic and statistical methods to enable computers to "learn" from the data made available to them, in other words, to improve performance in solving tasks without explicit programming for each one[5][19]. In order to automate the algorithm selection process, we use **supervised learning algorithms**, applied in our four defined configurations (configuration 1, configuration 2, configuration 3, configuration 4), using the two forms of representation of results: different models included in the supervised learning algorithms have been implemented to find the configuration with the best result.

Supervised Learning Algorithms

In supervised learning, the model is trained on labeled data and tested on unlabelled data [13][31]. The basic principle is to collect a data set, divide it into training and test sets, and preprocess the data. The features deduced

are inserted into an algorithm and the model is trained to record the features associated with each label. The model is then fed with test data, enabling it to predict outcomes by assigning expected labels.

The extracted features are fed into an algorithm and the model is trained to capture the features associated with each label. The model is then fed with test data, allowing it to predict outcomes by assigning expected labels. The next figure illustrate the supervised learning architecture in Figure6.1[13]:

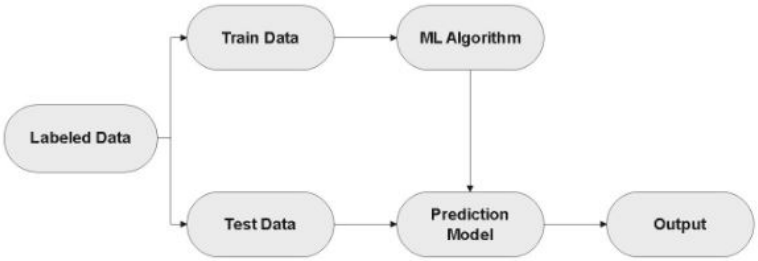

Figure 6.1: Architecture of Supervised Learning
[13]

The two types of supervised learning that exist are **Classification** and **Regression**, however in our case we focused only on the classification models described below:

- **Classification** Classification predicts unknown values or outputs from known values or inputs [31].To build a classification model, we first collect and pre-process the data. preprocessing removes noise and duplicates from the data. We use different methods to pre-process the data. We then divide the data into training and test sets using cross-validation. We train the model using class labels. We use Python's sci-kit learn package and its 'fit-transform(X, Y)' function to fit X (input data) to Y (labels) and build the classifier. We use binary and multi-label classification to predict the best configuration for our problems. We present the classification process [13] in figure 6.2:

Classification models were used in our case to predict which configuration is the best using an algorithm-based representation form for given instance problems.
Following classification models were evaluated:

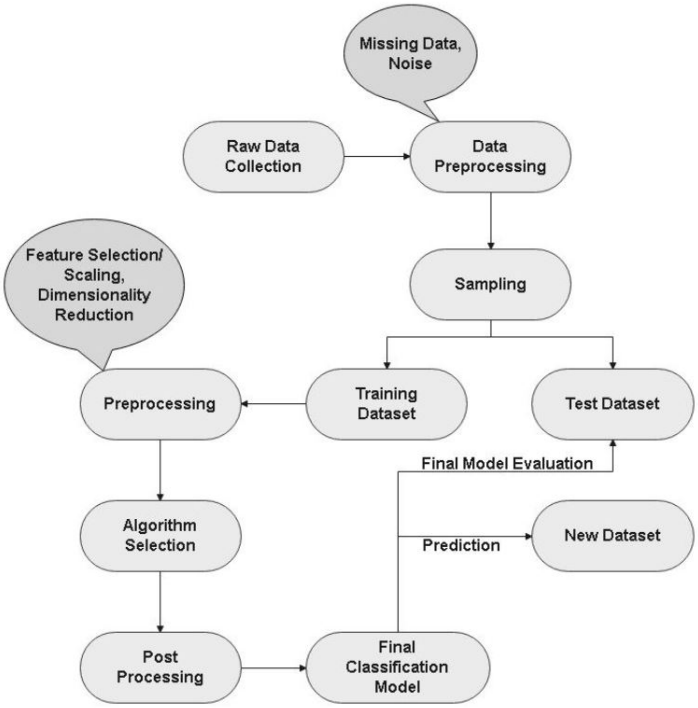

Figure 6.2: Process of Supervised Learning

1. Logistic Regression:

In the form of a binary model, a prediction is made about whether a data point belongs to one of the two categories, based on a probability threshold.In the case that the probability exceeds a value of 50 percent, the model anticipates the data point´s association with that category and outputs a positive binary value of 1. Conversely, if the probability is less than 50 percent, the model returns a zero value of 0.[15].the model anticipates the data point's association with that category and outputs a positive binary value of 1.

If the probability is greater than 50 percent, the model predicts that the data point belongs to that category and returns a non-zero positive binary value of 1 The logistic regression model works similarly to linear regression, but instead of producing a continuous output, it calculates the logistic function of the outcome [15]. The logistic function, denoted by $\sigma(\cdot)$, produces binary outcomes of 0 and 1. Its formula is shown below:[15]

$$\sigma(t) = \frac{1}{1 + exp^{-t}}$$

the general formula of the logistic regression is: [15]

$$\hat{y} = \begin{cases} 0 & \sigma(t) < 0.5 \\ 1 & \sigma(t) > 0.5 \end{cases}$$

Application of logistic regression In the case of our study, the objective is to predict whether configuration i is better (1) or not (0) based on the selected characteristics.
Data loading and Splitting The training.csv data file is first loaded (using pandas) and The dataset is divided into training and test sets, allocating 20 percent exclusively for testing:

```
import pandas as pn
from sklearn.model_selection import train_test_split
file = pn.read_csv("training.csv", sep=";")
train_set, test_set = train_test_split(file,
test_size=0.2,
random_state=42)
data = train_set.copy()
```

Selection of features Specific features such as: average duration, average resource requirements, available resources, average successors for training the model are selected according to the Latex code.

```
features = ["mean duration", "mean ressource demands",
"available ressources", "mean successors"]
X_train = data[features]
y_train = data["config i better"]
```

where x-train contains the feature values and y-train the target variable (config-i-better). We then instantiated a logistic regression model using the Python code below:

```
log_reg = LogisticRegression()
```

2. Stochastic Gradient Descent (SGD) Classifier

This is a linear classifier that applies regularised linear models with stochastic gradient descent (SGD) learning, which is very useful for implementing classification tasks and can be trained on large datasets under memory constraints. [35] Scikit-learn's sklearn.linear-model.SGDClassifier class was used to generate the SGD classifier. It supports different loss functions such as hinge,

log-loss, modified Huber, hinge squared, and perceptron.The classifier´s behavior can be influenced through the loss[35]. The classifier's behavior can be influenced through the loss parameter, which, can be aligned with a linear support vector machine(SVM). Different regularisation terms, denoted L1, L2, and Elastic Net, are included in the SGD class of classifiers. The regularization term introduces a penalty to the loss function, causing a reduction in model parameters toward the zero vector. This can be achieved using the squared Euclidean norm (L2), the absolute norm (L1), or a combination of both norms (L1 and L2), known as Elastic Net [35].

- **Application of SGD Classifier**

Importing the SGD Classifier Model

```
from sklearn.linear_model import SGDClassifier
```

Creating an Instance of SGD Classifier

```
sgd_class = SGDClassifier(random_state=42)
```

An instance of the SGD classifier is generated with a specified random seed (random-state=42) to ensure reproducibility. **Printing Qualities Measures**

```
print_quality_measures(sgd_class, "SGD Classifier")
```

The Print Quality Measures function is then called, and passes the SGD classifier instance and a label ("SGD Classifier"). This function performs cross-validation and prints various quality measures such as the confusion matrix, precision, recall and F1 score.

- **Training and Cross-Validation**

```
y_pred = cross_val_predict(sgd_class, X_train, y_train,
cv=3)
```

Scikit-learn's cross-val-predict function is used to perform cross-validation. It ensures that each instance is used exactly once as a test set and predicts for each data point. The predictions (y-pred) are then used to evaluate the performance of the model.

3. Decision Tree

A decision tree is a hierarchical model that comprises nodes and branches. The nodes represent characteristics, while the branches represent a value or condition associated with a node. Samples are classified by sorting them, starting from the root node, based on feature values [13]. At each sorting stage, the decision tree decides by selecting the most relevant alternatives. This is a simple, straightforward strategy that requires little data preprocessing and is easy to understand. Nevertheless, it isn't stable and can provide a complex tree structure [13]. His pseudocode is the following[13]:

Algorithm 4 Pseudo-Code of Decision Tree Algorithm [13]

1: Place the best attribute of the dataset at the root of the tree
2: Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for an attribute
3: Repeat step 1 and step 2 on each subset until you find leaf nodes in all the branches of the tree

- **Application of Decision Tree Model**

In this part each important factors used for the training of decision tree model can be detailed using the python code [6]. **importing Libraries:**

```
from sklearn.tree import DecisionTreeClassifier
```

The Decision Tree model is imported from scikit-learn's tree module. **Creating an Instance of DecisionTreeClassifier:**

```
decision_tree = DecisionTreeClassifier()
```

An instance of the DecisionTreeClassifier is created. This classifier is a part of the scikit-learn library and is used for decision tree-based classification. **Printing Quality Measures:**

```
print_quality_measures(decision_tree, "Decision Tree")
```

The print-quality-measures function is then called, passing the Decision Tree instance and a label ("Decision Tree"). This function performs cross-validation and prints the various quality measures for evaluating of the model.
Training and Cross-Validation:[3]

```
y_pred = cross_val_predict(decision_tree, X_train,
y_train,
cv=3)
```

Cross-validation is performed using scikit-learn's cross-val-predict function. This function provides predictions for each data point, ensuring that each instance is used exactly once as a test set. The predictions (y-pred) are then used to evaluate the performance of the model.

4. Random Forest Classifier

A random forest is characterized as a set of decision trees produced using the bagging method [3]. Rather than manually creating a BaggingClassifier and incorporating it into a DecisionTreeClassifier, it's advisable to employ the dedicated RandomForestClassifier class. This class acts as a classifier encompassing nearly all the hyperparameters of a DecisionTreeClassifier, offering control over the creation of trees. It also includes all the hyperparameters of a BaggingClassifier for ensemble management, with a few exceptions [15].

- **Application of Random Forest for the training**

Using the **scikit-learn** module, we imported the libraries required to use the model, then created an instance of RanomForestClassifier and finally printed out quality measures to evaluate the model.
Import libraries: The sci-kit-learn ensemble module is used to import the random forest classifier.

```
from sklearn.ensemble import RandomForestClassifier
```

Creation of an Instance of RandomForest Classifier

```
random_forest = RandomForestClassifier()
```

To instantiate the Random Forest classifier, a multitude of decision trees are concurrently trained via the Random Forest ensemble learning technique. The classifier then produces the class mode (classification) for each tree[35].
Print of Quality Measures

```
print_quality_measures(random_forest, "Random Forest")
```

5. Support Vector Machines(SVM)

The Support Vector Machine (SVM), as another beneficial machine learning model, manages linear classification tasks [28]. It also facilitates the execution of non-linear classification tasks[15]. SVMs are extensively utilized for

outlier detection, classification, and regression, handling both discrete and continuous cases. They depict features or events in an n-dimensional space, distinctly differentiating between various categories or classes. The SVM was implemented using Scikit-learn, and its pseudocode is outlined as follows[13]:

Algorithm 5 SVM Pseudo-Code

1: **procedure** SVM(Training and Test Data)
2: **In:** Determine training and test data
3: **Out:** Calculate accuracy
4: 1: Select optimal cost and gamma for SVM
5: 2: Implement SVM train step for each data point
6: 3: Implement SVM classify for test data points
7: 4: Repeat Steps 3 and 4 until Stop Condition
8: 5: Return calculated accuracy
9: **end procedure**

- **Application of Support Vector Machines SVM for the training**

Each step of the support vector machine application is explained as follows: **initialization step: SVC** stands for Support Vector Classification and is part of scikit-learn's SVM implementation. kernel="poly" indicates that we are using a polynomial kernel for SVM.

```
\begin{lstlisting}[language=Python]
svm_poly_kernel = SVC(kernel="poly", degree=3, coef0=1, C=5)
print_quality_measures(svm_poly_kernel, "Support Vector
Machines")
\end{lstlisting}
```

degree=3 represents the use of a polynomial kernel with degree 3. **coef0=1** indicates the independent term in the kernel function equation. **C=5** acts as the regularization parameter, managing the balance between minimizing training error and testing error. A larger C value promotes a hyperplane with a smaller margin. The **print-quality-measures** function prints various evaluation metrics for the SVM model [33]. The **cross-val-predict** function is utilized for generating cross-validated predictions. Under the section of **Training and Evaluation** and **print-quality-measures** functions invoked to output various evaluation metrics for the SVM model. The selection of a polynomial kernel of degree 3 implies that the decision boundary is represented as a cubic curve in the feature space.

For the evaluation of each model offering the best configuration with the best result for given instance problems, it was necessary to specify metrics for measuring the quality of each model mentioned.

- **Metrics of measuring the quality of supervised Learning models**

1. **confusionmatrix:** In our case, we use a metric like confusionmatrix to evaluate the predictive performance of each classifier model. The metric counts the data points where Category A is classified as Category B:[15] To calculate the confusion matrix value, we used a set of predictions that we compared with the target values. We first use the function **cross-val-predict()** to perform k-fold cross-validation. The idea behind this was to return the prediction n calculated for each test fold, providing a prediction for each point of data in the learning set. Predictions are values that each model generated, and our function cross-val-predict() provided a list of predictions where each element of the list corresponds to a prediction done for a particular sample in the test fold.
   After generating it and being notified as y-train-predict, we can calculate the confusion matrix needed by passing the target categories **x-train** and the predicted categories **y-train-predict**[15].

- **Application of confusion matrix** The confusion matrix is defined as a 2x2 square matrix that represents the model predictions and the current classes [8]. Each cell of the matrix corresponds to a specific count related to the model predictions[15]. In the given example in table 6.1 the confusion matrix (model, X-training, Y-training-prediction) is described as follows:

Actual		**Prediction**	
		Positive	**Negative**
Positive	**True Positive**	130	99
	False Negative	98	73

Table 6.1: Confusion Matrix Example

- **TP as true Positive** The instances correctly predicted as positive in the example with a value of 130, the instances for which the model predicted the positive class and the actual class was positive.

- **FP as false Positive** In the example given, the top right cell with a value of 99 represents the cases where the model predicted a positive class, but the actual class was negative.

- **FN as false Negative** The value of 98 in the bottom-left cell of the confusion matrix signifies instances where the model incorrectly predicted the negative class while the actual class was positive.

- **TN as True Negative** In this instance, the occurrences accurately predicted as negative are denoted by the bottom-right cell with a value of 73. at this point the model correctly predicted the negative class and the actual class was indeed negative.
  In our practical scenario, we evaluate two configurations (3 and 4). The confusion matrix reveals the instances where the model accurately predicted each configuration. Specifically, the interpretation is described as follows: Configuration 4 was correctly selected 130 times(the best choice); Configuration 3 was correctly selected 73 times; Configuration 4 was predicted 99 times, but Configuration 3 was the optimal choice; and Configuration 3 was predicted 98 times, but Configuration 3 was predicted 98 times, but Configuration 4 was the superior choice. regarding our actual case, where two configurations (3 and 4) are compared: It specifies how often the model correctly predicted the configuration using this case the matrix can be explained as follows: (130 times was the selection of Configuration 4, who was the best, 73 Configuration 3, 99 was Configuration 4 but configuration 3 was the best 98 was configuration 3 but Configuration 4 was the best.

  The confusion matrix used gives only one part of the information about the evaluation of a model, we needed other metrics like accuracy or relevance.

2. **Precision:** It is used for positive predictions, the formula is:

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

, where **TP** the Number of true positives and **FP** the number of false positives [7][15]. While achieving perfect accuracy (100 percent) in the case of a single positive prediction may seem desirable, this approach appears impractical. THE classifier, in such a scenario, would essentially disregard all positive data points except one.

Another metric like precision used was the sensibility.

3. **Sensibility** another name, known as the success rate or true positive rate (TPR): the proportion of positive data points after recognition by the classifier. Its formula is

$$\textbf{sensitivity} = \frac{TP}{TP + FN}$$

[7], with **TP** as the number of correct positives and **FN** the number of false Negatives [7][15].

4. **F1-Score:** The F1 score is determined by calculating the harmonic mean of precision and recall. Unlike the arithmetic mean, which considers all values equally, the harmonic mean gives more weight to the lower values. Therefore, a high F1 score can only be achieved if both precision and recall are high. The formula for the F1 score is defined as follows [15]:

$$\textbf{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}}$$

6.2 Practical Implementation

In this practical step, we will focus on the training of the best model selected from the five supervised learning models presented for algorithm selection from our portfolio of genetic algorithms, noting that the model selected and trained will result in the best configuration defined with the best result, this result coming from the case study j120 instance problems executed and also an average result that will allow us to make a comparison. We pay particular attention to genetic algorithms based on random keys and activity lists. This practical exploration aims to validate the effectiveness of our machine learning-based approach in selecting the optimal configuration including a representation form. First, we need a script that creates new data **results. csv** for training and validating each model. This script, in our case **Transformator. py**, reads the **results. csv** from an instance of the study case j120 and creates new machine learning data that can be used for training the machine learning model.

Features for the implementation These features calculated from each instance are used as training data for each model:

- **Mean duration** Calculate the average duration of all activities in the instance.

$$\mathbf{D} = \sum_{i=1}^{n} p_i$$

, with n last activity and p_i the duration of an activity i.

- **number of Activities** sums all activities included in the instance.

$$\mathbf{NA} = \sum_{i=1}^{n} 1$$

, with 1 who indicates the counting of activity p_i

- **Mean resource demands** Calculate the average resource demand across all activities in the instance. it sums the resource demands of all activities and divide by the total number of activities:

$$\mathbf{r} = \frac{\sum_{i=1}^{n} r_{ik}}{N}$$

with r_ik Resource requirement for an activity i of resource k.

- **Available resources:** It counts the total number of available resources. In the case study j120, there are four renewable resources

$$\mathbf{R_k} = 4$$

- **mean successors** Calculate the average number of successors for all activities in the instance, it sums the successors for each activity and divide by the total number of activities:

$$\mathbf{s} = \frac{\sum_{i=1}^{n} nbSuccessors_i}{N}$$

.

All these features are used for the training of the model and included in the training.csv obtained from the script Transformator.py.
In Application, to train our models, Transformator.py mentions the comparison between two given configurations, which will return the result 0 or 1 for better-config-i in a column, 1 in case the selected configuration is better than the other and 0 in the other case. with i as the enumerated number of the configuration in the parameter. In conclusion, for the training of our machine learning model, the Transformator.py script reads the results after execution of the different configurations and creates training data with six columns including all features described above which will be used by each classification model. The following table 6.2 show the related features needed for the training of the model.

Table 6.2: Features for the training of machine learning models

$Config - i - better$	**D**	**NA**	**r**	R_k	**s**
Values(1,0)					

An extract of twenty five values of training.csv resulting from transformator.py is presented in figure 6.5: The next important module for the training of supervised learning models is **Comparison. py** Includes five models from scikit-learn for comparison choosing the best model for the training. It includes modules like:

- **sklearn-model-selection** where the train-test-split , cross-val-predict, are imported.
- **sklearn-linear-model** where logisticRegression, SGDClassifier are imported.
- **sklearn-metrics** where metrics like confusion-matrix, precision-score, recall-score, and f1-score are imported.

6.2.1 Results of models and Observations

This part present results of the evaluation of all explored models based on the script for the machine learning. The test consists first of the comparison between configuration 2 and configuration 3, taken randomly as an example. The decision is based on two values included in training.csv using the index with values between 1 and 0, if the first configuration (configuration 2) is

selected and better than the next, the index value will take 1, else it will take the value 0, that means that the second configuration (configuration 3) compared is better than the first. The following twenty-five extracted results on 498 results from case study j120 including in figure 6.5 of the training data shows also features needed for the training of machine learning model and index value for the selected configuration.

For the comparison using features, the six diagrams show results level for

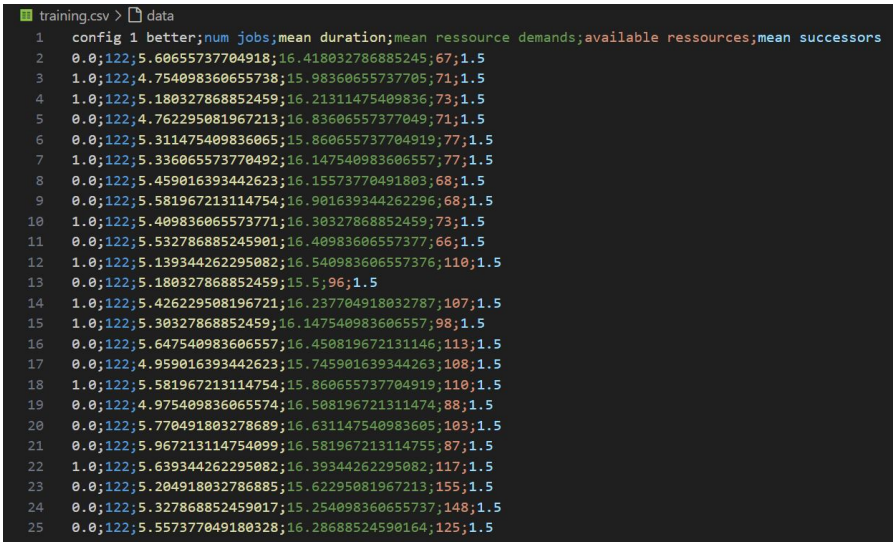

Figure 6.3: extracted results of 25 data in training.csv including features for Machine Learning models

the best configuration regarding index values(0 or 1),the mean duration, the available ressources, the mean number of jobs, the mean ressource demands and the mean successors and his evolution in figure 6.4.

The result of the evaluation of the five specified supervised learning models using the four metrics for configurations 1 to 4 can be seen in table 6.3 below:

Model	Conf. Matrix	Precision	Sensitivity	F1 Score
Logistic Regression	[226, 24], [132, 16]	0.4	0.10810810810810811	0.1702127659574468
SGD Classifier	[167, 83], [99, 49]	0.3712121212121212	0.3310810810810811	0.35000000000000003
Decision Tree	[158, 92], [95, 53]	0.36551724137931035	0.3581081081081081	0.36177474402730375
Random Forest	[181, 69], [103, 39]	0.3611111111111111	0.2635135135135135	0.3046875
Support Vector Machine	[191, 59], [107, 41]	0.41	0.27702702702702703	0.33064516129032256

Table 6.3: Comparison of Supervised Learning Models

As general observation, it emerges that the results of evaluation of each model

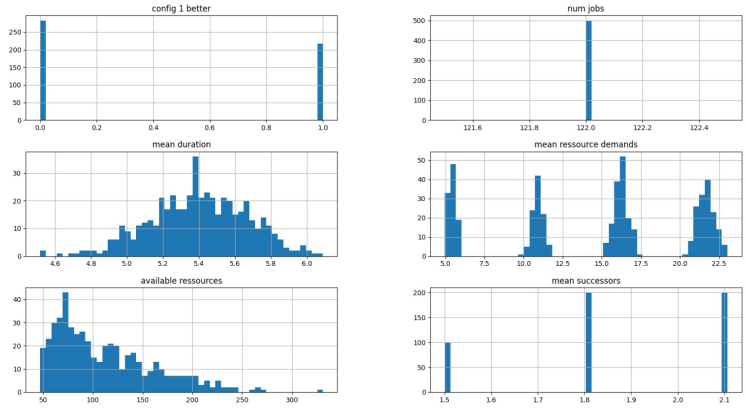

Figure 6.4: Visualisation of best configuration

do not look really good for any of the five models. The logistic regression performs particularly poorly. Decision trees, random forests, and Support Vector machines look the most promising. For our search, we decided to train one model and the selected model was the Decision Tree model for the specified case comparing configuration 3 and configuration 4 as basic examples.

6.2.2 Training of Decision Tree model and Observation

This part will focus exclusively on the training of the decision tree model as the selected model, the modules and parameters used, then the results obtained with prediction, followed by a general observation of the results obtained. In this subsection, we describe the training process of the decision tree model .
The objective is to optimize the model to obtain the best possible results by using a grid search to adjust the hyperparameters.

Modules Implementation

Three Scripts were developed for this task described as follow:

(a) **DecisionTree.py**: This script uses a grid search to establish the hyperparameters of the decision tree model using scikit-learn's `GridSearchCV` class.

(b) **PredictionTest.py**: is responsible for reading the obtained results, compares the times between two configurations, and utilizes the trained model to predict the optimal configuration and results.

(c) **SaveModel.py**: trains the decision tree model with the best parameters found, it saves it to a file and assesses its performance on test data.

Module Parameters

we first found the best hyperparameters for the decision tree responsible for the optimization of the results. These hyperparameters, generated for training the decision tree model, are presented as follows:

- **criterion:gini, entropy, log-loss, default=gini:** responsible for measuring the quality of a division. The criteria considered in the implementation are gini for Gini impurity and log-loss and entropy, both for Shannon information contribution[36].
- **splitter: "best", "random", default=" best":** this strategy is responsible for choosing the division at each node. The strategies supported are "best" for the best-split option and "random" for the best-random split option. In this case, "best" is used [36].
- **max depth:** The maximum depth limit of the tree. If not specified (None), nodes are expanded until all leaves are clean or until each leaf contains fewer samples than the number defined by `min-samples-split`. The default value for this parameter is None[36].
- **min samples split: int or float, default=2:** The minimum number of samples required to perform a split at an internal node. If `min-samples-split` is an integer, it directly represents the minimum number. On the other hand, if it is a floating-point number, `min-samples-split` is interpreted as a fraction, and the minimum number of samples for each division is set to `ceil(min-samples-split * n-samples)`, fixed at 16[36].
- **min samples leaf: int or float, default=1:** The minimum number of samples required to form a leaf node. A split point at any depth will only be considered if it results in at least `min leaf samples` learning samples in each of the left and right branches. If `min leaf samples` is an integer, it directly represents the minimum number. However, if it is a floating number, `min-samples-leaf` is interpreted as a frac-

tion, and the minimum number of samples for each node is set to `this(min-samples-leaf * n-samples)`.

The values given were determined using the grid search and it was following [35]:

- **Criterion:** gini
- **Splitter:** random
- **Max Depth:** 32
- **Min Samples Split:** 4
- **Min Samples Leaf:** 1

The DecissionTree model can be generated with the parameters above, trained with the **training data set**, saved in a file and checked with the **test data set**, which can be set aside. This is done by the "SaveModel.py" program, which saves the trained model in the "decission-tree-model.pkl" file.

Results achieved

The results of the decision tree model regarding his qualities were evaluated using the four specified metrics described in the last section. These results were obtained using training data between configurations 3 and 4:

- **Confusion matrix:** [[177 73] [91 57]]
- **Precision:** 0.43846153846153846
- **Sensibility:**0.38513513513513514
- **F1 Score:** 0.41007194244604317

These results show how well the decision tree model performs on training data.

Observations

The results obtained with the test data show a slight decrease in performance, which is expected. However, using the model to predict the best-performing configuration or algorithm contained in the portfolio among algorithms 3 and 4 showed good results as presented in the following result, confirming the effectiveness of the trained model, even the meantime prediction can be accepted. The most promised algorithm or configuration in the case of the

application was configuration 4 using the activity-list-based representation form with a mean time of 142.94578313253012 against 142.97991967871485 for the configuration 3, but the most important result is the predicted result with a mean time prediction of 142.52008032128515, regardings this result we can conclude that the model can predict good results by the automatic selection of the best algorithm or configuration as than solving only one algorithm separately

```
eit/Implementierung Masterarbeit/Machine Learning/PredictionTest.py"
start index: 9
end index: 506
mean time Config 3:  142.94578313253012
mean time Config 4:  142.97991967871485
C:\Users\Dell\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have valid feat
ure names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
mean time prediction:  142.52008032128515
```

Figure 6.5: results of prediction for the decision tree for configuration 3 and 4

Overall, the decision tree model appears to generalize well and is capable of making reliable predictions on unknown data.

Implementation Overview

This chapter details the steps leading to the answer to the question posed in our study, namely whether automatic algorithm selection can improve performance over the use of a single algorithm to solve the problem of RCPSP. The practical application of the algorithmic components and concepts involved in the resource constrained Project Scheduling Problem was followed by several important stages, each of which adds value to the general understanding and improvement of resource-constrained project planning. The application of the concept of automatic algorithm selection based on applied intelligent learning was then a very important part. To achieve these detailed objectives, the first step was to focus on the concept of the resource-constrained project planning problem, illustrating the fundamental features such as identifiers, resources, resource utilization, and capacities. This provided the basis for understanding some of the challenges that need to be addressed regarding resource constraints in project planning. We then set out to highlight the mathematical problem to which we should pay particular attention, that of minimizing the time taken to implement a scheduling problem, given time and resource constraints. We also introduced several concepts and models involved, including the metra potential method or MPM and the resource profile to represent a problem instance. To understand the concept of problem-solving, we introduced the Tripel Algorithm, which is essential for calculating the earliest date (ES) and the latest date (LS). From the results obtained, also known as the distance matrix, we were able to state the algorithm for generating a serial schedule, taking into account the priority rules of the LST rule.
Having completed this essential part of our study, we focused on the introduction and application of the Genetic Algorithm as a methodology for dealing with resource-constrained, high-intensity project scheduling problems to optimize results, with particular emphasis on optimizing project completion time, known as Makespan. In the same vein, we described the building blocks of the Genetic Algorithm, specifying our focus on two main models of result representation, the point of coding, of which two were the most impor-

tant, the representation based on activity lists, and the representation based on random keys, then other important elements of the Genetic Algorithm were the selection methods, the crossover and mutation operators. A case study based on an instance was the subject of the application of the genetic algorithm, and the concept of temporal planning was also explained.

In the practical implementation of the genetic algorithm, we had to define the initial population, and evaluate the best individuals using the fitness function, followed by selection, crossover, mutation, and generation of the best time or duration, taking into account that the best possible optimal duration is the one that gives us the minimum result. The results and the analysis following a resolution of the instances contained in the class of study j120 after the definition of four configurations including the different forms of representation of the results allowed us to evaluate the practical aspects of the implementation of the portfolio of genetic algorithm including the environment of programming, the language of programming, the environment of execution and also the parameters, the instances of resources for the portfolio, which was always the framework of study j120. In the last part of our work, it was a question of optimizing our results in the solution of the problems of instance through an automatic selection of algorithm, for that we evaluated based on comparison of two configurations of genetic algorithm five models of supervised learning model of intelligent learning, in which we chose following the units of measurement of the models, the model of a decision tree that was trained and the results obtained following implementation of the model.

Outcomes

The detailed problem description and literature review have contributed to a comprehensive understanding of the challenges posed by resource constraints in project scheduling. The mathematical formulation of RCPSP provides a solid foundation for subsequent algorithmic approaches. Genetic Algorithms have been highly effective in generating optimal project schedules under resource constraints. The encoding and decoding techniques, initial population generation, and evolutionary processes such as selection, crossover, and mutation demonstrated robustness in finding solutions for complex scheduling scenarios. The experimental setup and execution of all created configurations or algorithms using the Genetic algorithm approach and his parameters were validated through case studies, showcasing the algorithm's adaptability to diverse project scheduling scenarios. The time-permissible schedule concept proved instrumental in aligning scheduling solutions with real-world project constraints. Machine learning-based optimization for algorithm selection demonstrated highly promising results and gave us an answer about the optimization of solution through selection of correspondent algorithm for a given problem instance. The portfolio of Genetic Algorithms, coupled with supervised learning, showcased improved decision-making capabilities in selecting the most suitable algorithm for specific instances. The practical implementation of the machine learning model training the decision tree model provided a streamlined approach to choosing the optimal algorithm based on the characteristics of the scheduling problem through compared configurations or algorithms created. Results from models and observations emphasized the effectiveness of this approach in enhancing scheduling solutions.

Anhang

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind, und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde. Ich stimme hiermit außerdem einer softwaregestützten Plagiatsprüfung zu.
Die benuzten Hilfsmitteln waren:

- **ChatGPT**; für das Schreiben in Latexcode bei der Erstellung von Tabellen, Algorithmen und Abbildungen sowie die mathematischen Formeln . Bei der Suche, die zum Verständnis der Konzepte des Themas zählt.
- **zeroGPT**
- **Microsoft Bing COPILOT**
- **Deepl Write**
- **Paperpal**
- **Grammarly and Grammarly Plagiarism checker**
- **Scribd Plagiarism checker**
- **Overleaf for latex writing**

Clausthal-Zellerfeld, den 05.02.2024,

Acknowledgments

During our work there are people who help us to achieve our goal. These indispensable supporters, both in practice and in the drafting of this work, deserve our thanks and recognition. Firstly, we are grateful to those people, especially our friends, family, and others who have taken the time to support us with ideas, presence, and advice, to help us achieve the ultimate goal of this work. we thank generally our academic supervisor **Mrs Lena Sophie Wohlert** and our professor **Prof. Dr. Jürgen Zimmermann**, who have supported and followed us during this crucial period in our work.

Bibliography

[1] David W Aha. Generalizing from case studies: A case study. In *Machine Learning Proceedings 1992*, pages 1–10. Elsevier, 1992.

[2] Christian Artigues, Roel Leus, and Fabrice Talla Nobibon. Robust optimization for the resource-constrained project scheduling problem with duration uncertainty. *Handbook on Project Management and Scheduling Vol. 2*, pages 875–908, 2015.

[3] Géron Aurélien. Hands-on machine learning with scikit-learn & tensorflow. *Geron Aurelien*, 134:145–150, 2017.

[4] Önder Halis Bettemir. Optimization of time-cost-resource trade-off problems in project scheduling using meta-heuristic algorithms. 2009.

[5] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[6] Daniele Bonacorsi, Luigi Guiducci, Carlo Battilana, and Tommaso Diotalevi. Cms level-1 trigger muon momentum assignment with machine learning. *Journal of Instrumentation*, 16(2):P02007, 2021.

[7] Fabian Bong and Joanna Mills. Predicting the outcome of nhl games. *Journal of Sports Analytics*, 7(4):233–244, 2021.

[8] Caren Brinckmann. *The Kiel corpus of read speech as a resource for speech synthesis*. PhD thesis, Citeseer, 2004.

[9] Naggula Chinna. Genetic algorithm. 10 Jan 2022. Accessed on 22 Dec 2023.

[10] Diane J Cook and R Craig Varnell. Maximizing the benefits of parallel search using machine learning. In *AAAI/IAAI*, pages 559–564, 1997.

[11] Jim Demmel, Jack Dongarra, Victor Eijkhout, Erika Fuentes, Antoine Petitet, Rich Vuduc, R Clint Whaley, and Katherine Yelick. Self-adapting linear algebra algorithms and software. *Proceedings of the IEEE*, 93(2):293–312, 2005.

[12] John Doe. Python advantages: A comprehensive exploration of python's benefits in software development. *Python Journal*, 1(1):1–10, 2022. This article provides an in-depth analysis of the advantages of using Python in various aspects of software development.

[13] Salim Dridi. Supervised learning-a systematic literature review. *Journal of Machine Learning Research. OSF Preprints*, 22(5):1234–1256, 2021.

[14] Stephanie Forrest. Genetic algorithms. *ACM computing surveys (CSUR)*, 28(1):77–80, 1996.

[15] Aurélien Géron. *Praxiseinstieg Machine Learning mit Scikit-Learn und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme*. O'Reilly, 2018.

[16] SM Hardi, M Zarlis, S Effendi, and Maya Silvi Lydia. Taxonomy genetic algorithm for implementation partially mapped crossover in travelling salesman problem. In *Journal of Physics: Conference Series*, volume 1641, page 012104. IOP Publishing, 2020.

[17] Sönke Hartmann. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL)*, 45(7):733–750, 1998.

[18] Frank Hutter, Youssef Hamadi, Holger H Hoos, and Kevin Leyton-Brown. Performance prediction and automated tuning of randomized and parametric algorithms. In *Principles and Practice of Constraint Programming-CP 2006: 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006. Proceedings 12*, pages 213–228. Springer, 2006.

[19] Maryam Karimi-Mamaghan, Mehrdad Mohammadi, Patrick Meyer, Amir Mohammad Karimi-Mamaghan, and El-Ghazali Talbi. Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, 296(2):393–422, 2022.

[20] Marcin Klimek. A genetic algorithm for the project scheduling with the resource constraints. *Annales Universitatis Mariae Curie-Skłodowska. Sectio AI, Informatica*, 10(1), 2010.

[21] Piotr Klimek. Genetic algorithm for scheduling projects with resource constraints. *Journal of Theoretical and Applied Computer Science*, 4(1):33–44, 2010.

[22] Rainer Kolisch and Sönke Hartmann. *Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis*. Springer, 1999.

[23] Rainer Kolisch, Arno Sprecher, and Andreas Drexl. Characterization and generation of a general class of resource-constrained project scheduling problems. *Management science*, 41(10):1693–1703, 1995.

[24] Lars Kotthoff. Algorithm selection for combinatorial search problems: A survey. *Data mining and constraint programming: Foundations of a cross-disciplinary approach*, pages 149–190, 2016.

[25] Haibo Li, Shaoyuan Weng, Juncheng Tong, Ting He, Wenyun Chen, Mengmeng Sun, and Yingtong Shen. Composition of resource-service chain based on evolutionary algorithm in distributed cloud manufacturing systems. *Ieee Access*, 8:19911–19920, 2020.

[26] Sakuntala Mahapatra. Genetic algorithm.ppt. `https://www.scribd.com/document/490135901/Genetic-Algorithm-ppt`, 2020. Accessed on 12 Dec 2023.

[27] Tarek M Mahmoud. A genetic and simulated annealing based algorithms for solving the flow assignment problem in computer networks. *International Journal of Computer and Information Engineering*, 1(3):472–478, 2007.

[28] D Jeya Mala. *Integrating the Internet of Things into software engineering practices*. IGI Global, 2019.

[29] Rafael Mart, Panos M Pardalos, and Mauricio GC Resende. *Handbook of heuristics*. Springer Publishing Company, Incorporated, 2018.

[30] Tommy Messelis and Patrick De Causmaecker. An automatic algorithm selection approach for the multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 233(3):511–528, 2014.

[31] Iqbal Muhammad and Zhu Yan. Supervised machine learning approaches: A survey. *ICTACT Journal on Soft Computing*, 5(3), 2015.

[32] Mladen Nikolić, Filip Marić, and Predrag Janičić. Instance-based selection of policies for sat solvers. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 326–340. Springer, 2009.

[33] Hafsa Ouchra, Abdessamad Belangour, and Allae Erraissi. Comparison of machine learning methods for satellite image classification: A case study of casablanca using landsat imagery and google earth engine. *Journal of Environmental & Earth Sciences*, 5(2):118–134, 2023.

[34] Hela Ouerfelli and Abdelaziz Dammak. The genetic algorithm with two point crossover to solve the resource-constrained project scheduling problems. In *2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO)*, pages 1–4. IEEE, 2013.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. Scikit-learn: Machine learning in Python, 2011. Software.

[36] Fabian Pedregosa, Ga ël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[37] John R Rice. The algorithm selection problem. In *Advances in computers*, volume 15, pages 65–118. Elsevier, 1976.

[38] L. Santos, R. Santos, R. Barbosa, and L. Santos. Adaptive biased random-key genetic algorithm with local search for the capacitated centered clustering problem. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2021.

[39] Andreas Schutt, Thibaut Feydy, Peter J. Stuckey, and Mark G. Wallace. Solving the resource constrained project scheduling problem with generalized precedences by lazy clause generation, 2010.

[40] MA Shouman, MS Ibrahim, M Khater, and AA Forgani. Genetic algorithm constraint project scheduling. *Alexandria Engineering Journal*, 45(3):289–298, 2006.

[41] Vicente Valls, Francisco Ballestin, and Sacramento Quintanilla. A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European journal of operational research*, 185(2):495–508, 2008.

[42] SG Varun Kumar and R Panneerselvam. A study of crossover operators for genetic algorithms to solve vrp and its variants and new sinusoidal motion crossover operator. *International Journal of Computational Intelligence Research*, 13(7):1717–1733, 2017.

[43] Lena Sophie Wohlert and Jürgen Zimmermann. Resource overload problems with tardiness penalty: structural properties and solution approaches. *Annals of Operations Research*, pages 1–22, 2024.

[44] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[45] Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *Journal of artificial intelligence research*, 32:565–606, 2008.

[46] Hua Zhang, Hao Xu, and Wuliang Peng. A genetic algorithm for solving rcpsp. In *2008 international symposium on computer science and computational technology*, volume 2, pages 246–249. IEEE, 2008.

[47] Jürgen Zimmermann, Christoph Stark, and Julia Rieck. *Projektplanung: Modelle, Methoden, Management*. Springer-Verlag, 2006.