

A learnheuristic method for solving resource constrained project scheduling problem

Behnam Jahani^{a*} and Mohammad Amin Adibi^a

^aDepartment of Industrial Engineering, Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran

ABSTRACT

Article history:

Received May 20, 2025

Received in revised format May 30, 2025

Accepted July 8 2025

Available online

July 8 2025

Keywords:

Resource constrained project scheduling problem (RCPSP)

Learnheuristic

Decision Tree

Genetic Algorithm

Project scheduling in resource-constrained mode is one of the most important issues in the field of project management. The main philosophy of this problem is to use less resources while respecting the resource limit to complete the project in a shorter time although other goals can be considered. When a very large amount of data is generated by the meta-heuristic algorithm and there are many variables involved in solving the problem, no other algorithm or technique is able to analyze the output. For this purpose, learnheuristics have the ability to use combined metaheuristics and machine learning tools with high accuracy and in less time to analyze data. The primary purpose of this research is to combine machine learning and genetic algorithms to reduce the project completion time which can lead to a reduction in the cost of the project. Due to the population-based nature of the problem a large amount of initial population was generated. In order to convert the generated schedules into feasible ones, a repair strategy was used. A data matrix was created to import data into the ML model. After specifying the training and testing settings of the model, the decision tree was used to analyze the data of the problem, then its output was applied to the initial population using the displacement or relocation procedure. This manipulated population is given to Genetic Algorithm (GA) and continues until a certain iteration. j60data on the PSPLIB website was used to evaluate the suggested approach. The findings indicate that the implemented approach has improved by 21.75% compared to the normal GA. This improvement means that a better solution could be achieved in less time with fewer calls.

© 2025 by the authors; licensee Growing Science, Canada.

1. Introduction

Scheduling activities while taking precedence relations and resource constraints into account is a common challenge in project scheduling, known as the resource-constrained project scheduling problem (RCPSP) (Li et al., 2024). The objective of the RCPSP is minimizing project completion time (C_{max}) although various other time-dependent goals have been also. The primary RCPSP demands that all activities be carried out without delay, as soon as they are initiated but cannot be stopped at any time and must continue to process them until completion (Hartmann & Briskorn 2021). Renewable resources have been considered to be part of the activities in most articles dealing with scheduling problems. It means that resource constraints have been frequent and when the activities are started, at the end of the process, a certain amount of these types of resources is taken up and recovered (Habibi et al., 2018; Afshar et al., 2022; Guo, 2021; Hua et al., 2022).

Industries with projects engaging various activities and restricted resources will find the RCPSP particularly pertinent (Khajesaedi et al., 2025). The exact methods, heuristics and metaheuristic approaches are three ways to solve the RCPSP. For exact methods, various researches have shown that they are inappropriate for solving high-volume data in a short time. Also, when many variables are involved in solving or analyzing a problem, accurate and heuristic methods do not work well. In addition, a number of researchers have shown that metaheuristic approaches are superior to heuristic approaches (Sebt et al., 2022). Heuristics and metaheuristics have been put forth to get near-optimal or optimal solutions with limited resources in an acceptable amount of time, especially for larger-scale problems. The best current methods of resolving the RCPSP are

* Corresponding author

E-mail address behnamjahaniacademic@gmail.com (B. Jahani)

therefore metaheuristic approaches. The two major planning methods are the critical path (CPM) and program evaluation and review technique (PERT). In recent decades, the CPM has been widely used for project planning, especially for key activities, sequential connections and temporal variables. But the CPM and PERT do not typically expect unlimited resources (Bettemir & Sonmez 2015; Cheng et al., 2014). Resources are generally scarce in engineering practice, (given to the nature of the project environment) which means that there is usually insufficient supply of resources to meet the needs. Therefore, it is common for CPM and PERT schedules to require optimization due to resource constraints on the project. Due to the sensitive and important nature of the RCPSP problem in different industries, various studies have been conducted on it: In the construction, manufacturing, aeronautics, and software development sectors it has been extensively studied. There are different types of resources in this problem: Renewable, nonrenewable, partially renewable and doubly constrained resources. Resources in RCPSP are renewable resources including work force, machinery, and equipment (Kolisch & Padman 2001). It has been shown that RCPSP is an NP-hard problem (Blazewicz et al., 1983).

According to the above, when we have a large number of data records and the number of activities and relationships between them is more, faster and smarter tools are needed. For this purpose, machine learning (ML) as one of the adaptable tools can be used in different fields in a very effective way. In the area of data analysis and computing, AI has grown rapidly over the past few years, in particular with regard to ML, which enables programs to be smart in most cases (Sarker, 2021). ML usually allows systems to learn and enhance their experience automatically, with no specific planning and in the fourth industrial revolution, it's generally considered to be the most popular new technology. In addition to the various applications of ML, there have been recent studies in the field of scheduling and ML tools. The decision tree (DT) method is a well-established, nonparametric supervised learning methodology (Quinlan JR. 1993). For classification and regression tasks, DT skills are used (Pedergosa et al., 2011). For DT algorithms, C4.5, ID3 and CART are widely used. The decision tree is capable of recognizing the trend and pattern of the data in question. They are therefore a good tool for detecting the population (in this case are schedules) created by metaheuristic algorithms or models. Combining machine learning and metaheuristics to improve efficiency of optimization processes is known as learnheuristics in the literature.

Our goal is to present a learnheuristics method to use decision tree to improve scheduling methods in order to find a new way to use ML in solving scheduling problems. As a result, in this research a very large starting population is created. The parameters of the network and resources together with the generated population from the input of our machine learning model. The fitness function or the completion time of each schedule is the output of the model. This table is called the data matrix table. The contribution of this research is combining DT and genetic algorithm (GA) to analyze the generated initial population. The model is trained and through the decision tree the data is analyzed. The output extracted from the model, which is called the rules, is coded to be applied to the primary population. We have named this method as the relocation method (which is the relocating of the genes in every schedule in a way that every predecessor of every activity is not violated). After that, every initial population that is produced has the desired characteristics, which is the low value of the fitness function (C_{max}). By giving this manipulated population to the GA in different iterations, the probability of reaching better solutions with fewer calls of the fitness function and in less time increases significantly. This results, on the other hand, leads to a reduction in the cost of the project, especially in repetitive activities. In the next section, there is a comprehensive literature review about this topic. In section 3 the RCPSP problem description is presented. In section 4, the GA algorithm operators used in this research are explained. In section 5 the methodology of this research which includes implementation of machine learning and decision tree is explained and the dataset is detailed. In section 6, computational tests are shown and in section 7 the results and comparison with the normal GA algorithm and future research are presented.

2. Literature Review

First, two studies using precise methods are mentioned. Then the innovative method is mentioned. After these, the focus will be on the studies that have used Advanced heuristic-based techniques to resolve the problem. Then the latest and closest researches that have used the combination of meta-heuristic algorithms and artificial intelligence to solve their problem will be explained. Finally, the difference between the approach of the present study and the previous studies and its innovation is expressed. One of the latest studies in RCPSP with exact methods is the Liu et al. (2023) research. In order to solve a project scheduling challenge involving single-capacity resources and transfer durations, Liu et al. (2023) suggest using a branch-and-bound algorithm. Every resource in the issue is distinct, and there is no way to minimize the time spent moving resources between tasks. The goal is to shorten the overall project duration. by identifying a workable solution that includes a plan for resource transfer and a vector of activity start times.

Their branch-and-bound algorithm employs a planning technique for the purpose of assessing commencement time of the subsequent task, ensuring it is viable and cannot begin before the initiation time of the previous planned task, and a branching strategy to explore all suitable activities to be assigned at each point. The outcomes demonstrate that their precise algorithm outperforms both CPLEX and CP Optimizer when solving the problem using the existing mathematical models. By providing a simple and efficient R&S approach for solving RCPSP, Etmianiesfahani et al. (2022) study adds to the body of knowledge already available on the relax-and-solve. Within a heuristic framework, our R&S generates and optimizes schedules using the CPLEX CP optimizer as an optimization solver. By using forward-backward passes, we are able to further enhance the algorithm's performance. Our heuristic yields strong results and surpasses cutting-edge methods for solving the RCPSP, according to the outcomes of evaluating the algorithms on 1560 benchmark cases from the popular PSPLIB.

As we can see, heuristic and metaheuristic methods are using the same concepts and methods in some way. However, as we explain in the introduction section, exact and heuristic methods are hard to implement and cannot solve the problem in a short period of time in comparison with the metaheuristic algorithms. Although they can reach the optimal solution but with a large amount of data and the nature of the problem when involved with many variables the analysis of relation between variables is impossible or hard to achieve and the chance to trap in a local optimum is high. According to this we concentrated on metaheuristic algorithms. Now some of the most related studies that have been using metaheuristic algorithms are these. Shuvo et al. (2023) are putting forth in this paper a hybrid metaheuristic method called CRO-GA, which combines chemical reaction optimization (CRO) and genetic algorithm (GA). In order to determine the answers, they have restructured the fundamental operators of GA and CRO. for the purpose of adaptation with GA, CRO uses an additional operator known as the priority-based selection operator. They compare our proposed method with other relevant techniques, including adaptive particle swarm optimization (A-PSO), multi-agent optimization algorithm (MAOA), artificial bee colony (ABC), and genetic algorithm (GA), which represent the current leading approaches for the RCPSP. The findings from the experiment demonstrate that, when solving RCPSP in less computational time, their suggested methodology outperforms other current algorithms.

As we can see in this research when GA algorithm combines with another method generally the results improve more. One of the key project scheduling issues is the multimode resource-constrained project scheduling problem (MRCPS), which is the subject of Sebt et al. (2022) paper. To address this, a new local search technique is proposed along with a genetic algorithm (GA) to solve the multimode resource-constrained project scheduling problem (MRCPS). The multimode serial schedule generation scheme (MSSGS) is used as the interpretation process, and encryption is done using the Activity Mode List (AML). The recommended local search seeks to reduce the total duration of the project by making the best use of its current nonrenewable resources. The neighborhood method assigns the exhaustible resources that are not being used in every valid solution for the activities that have a reduced total float for this reason. The efficiency of the suggested algorithm for MRCPS solution. is confirmed by contrasting the outcomes of the suggested approach with those of alternative methods utilizing the J20 collection in the Project Scheduling Problem Library (PSPLIB). Roy et al. (2023) study modifies the widely used Firefly Algorithm (FA), a Swarm Intelligence (SI) metaheuristic, to solve the RCPSP. A discrete framework has been provided for the intensity-based firefly movement by applying a 2-point crossover. Furthermore, a swap-based mutation is added to lessen the likelihood that the solution will become stuck in the local optima. Extensive experiments on well-known benchmark instances have been used to thoroughly evaluate the performance of the proposed research. The approach's competitiveness was demonstrated through comparison with seminal works.

Z. Liu et al. (2022) paper proposes an improved genetic algorithm based on time window decomposition. In order to improve population diversity, three derivation techniques are used. The search efficiency is increased by using destructive lower bounds and the sampling size approach for distributing. The PSPLIB computational experiments demonstrate that the suggested method is strong in addressing two practical scenarios. and more effective than relying solely on the decomposition mechanism. This work shows that there may be benefits to altering the search subspaces on a regular basis. In this research the exploitation of metaheuristic algorithms reduces the chance of solutions trapped in local optima. These findings could be helpful when examining RCPSP with different evolutionary algorithms in the future. By utilizing machine learning techniques to flexibly determine the sampling times for each individual, it may be possible to get some other better results. According to the above research, the meaningful and convenient combination of meta-heuristic algorithms with other methods and their high speed in solving problems has turned them into unique tools for use. When meta-heuristic algorithms are combined with other methods, the improvement rate of the problem's solution increases significantly. Yet Current research tends to concentrate on developing appropriate algorithms to solve different types of scheduling problems, but it does not identify possible scheduling guidelines in these near-perfect or perfect outcomes—specifically, potential inherent connections between attributes related to activity sequence planning.

Large-scale data is stored with valuable information that can be retrieved through analysis and interpretation using data mining (DM). In order to approximate efficient solutions to resource-constrained project scheduling problems, the aim of Xie et al. (2023) paper is to apply DM to discover scheduling concepts. These rules, which have a very reduced time complexity and facilitate immediate decision-making to enhance planning and scheduling, do not call for search or simulation. His paper suggests a novel method that combines the genetic algorithm (GA) and DM technology to solve the resource-constrained project scheduling problem. To be more precise, the GA is used to produce a variety of ideal project scheduling plans. Next, the C4.5 decision tree (DT) is utilized to extract useful information from these schemes in order to better anticipate and resolve future scheduling issues. In order to identify the scheduling rule set that will shorten the total duration, the authors of this study examine and derive insights from numerous scheduling methods using GA and DM technologies. The proposed DT classification model's application effect is confirmed by testing the scheduling rules' validity on the J30, J60, and J120 datasets within PSPLIB. The outcomes demonstrate that DT can easily match GA's superior performance for scheduling problems of various sizes.

Because priority rules have well-established benefits like being quick to implement, simple to use, and easy on the eyes, many commercial software tools use them to schedule projects with limited resources. Furthermore, despite the fact that a large number of research studies compare various priority rules, managers frequently are unable to determine which rule is best for their particular project and are forced to choose a random priority rule, with the hope for a favorable outcome. In order to categorize and identify the optimal priority rule for RCPSP, Vanhoucke et al. (2021) paper presents a decision tree approach.

Two classification models are used in the study to assign project metrics to the priority rule's efficiency. These models allow for the prediction of each priority rule's performance, this information is afterwards employed to autonomously choose the most effective ranking criterion for the specific project., using values from resource and network indicators that are known. The performance of the recently proposed classification models is assessed through a series of computational experiments that make use of the most prominent priority rules highlighted in the scholarly works. The tests demonstrate that multi-label classification models can surpass the typical capability achieved by employing each individual priority rule. when comparing their performance with multi-class classification models. It will be claimed that with no modifications to the methodology employed in this study, this method is easily transferable to any extension of the RCPSP.

The standard single mode RCPSP is addressed in Golab et al. (2023) paper using a convolutional neural network approach. Unlike other approaches like metaheuristics, this algorithm has the benefit of not requiring a large number of populations or solutions to be generated. This study utilizes serial schedule generation scheme (SSGS) for arranging the project tasks through a developed convolutional neural network (CNN) as a mechanism to choose a suitable priority rule to exclude a potential activity. These eight project metrics—network complexity, resource factor, resource strength, average work per activity, etc.—are what the evolved CNN learns based on. Each stage of the project planning process involves recalculating the aforementioned parameters, which serve as the input provided to the network. The outputs of the developed neural system are also included in the network's priority rules. To exclude an activity from the list of qualified activities, the network can therefore automatically choose a suitable priority rule following the learning process. According to the specified project constraints, the algorithm can thus schedule every activity for the project. Finally, they compare the performance of the MLFNN approach with that of conventional metaheuristics and the Convolutional Neural Network (CNN) approach, employing established reference point problems from PSPLIB. Table 1 summarizes the previously mentioned research and this study with respect to the problem type, problem objective and the technique.

Table 1
Literature review

Reference	Type	Objective	Exact	Heuristic	Metaheuristic	Decision tree	ANN
Liu et al. (2023)	RCPSP	Single-Minimizing project completion time	✓				
Etminaniesfahani et al. (2022)	RCPSP	Single- by providing R&S approach generating and optimizing schedules using the CPLEX CP optimizer	✓				
Shuvo et al. (2023)	RCPSP	Single- an additional operator known as the priority-based selection operator			✓		
Sebt et al. (2022)	MRCPSP	Single-Minimizing project completion time			✓		
Roy et al. (2023)	RCPSP	Single-Modifying the widely used of firefly and swarm intelligence algorithms with a discrete approach			✓		
Z. Liu et al. (2022)	RCPSP	Single-improving GA by the concept of time windows decomposition			✓		
Xie et al. (2023)	RCPSP	Single-Minimizing the project completion time			✓	✓	
Vanhoucke et al. (2021)	RCPSP	Single-Identifying top-performing priority rule				✓	
Golab et al. (2023)	RCPSP	Single-Scheduling activities					✓
This paper	RCPSP	Single- combining GA and machine learning to reduce project completion time			✓	✓	✓

According to our understanding, despite a variety of research on the RCPSP in various optimization models, none of them have tackled the problem using learnheuristic to generate better solutions. Therefore, by using metaheuristic algorithms, this paper aims to close the gap in the literature by referring to or taking into consideration machine learning in the form of learnheuristic.

3. Problem Description

A conceptual project is considered in the RCPSP formulation. There are n activities in total, and to complete the project, every task must be executed without fail. Additionally, the project's start and finish are represented by the dummy activities 0 and $n + 1$. d_j , where $d_0 = d_{n+1} = 0$, represents the duration of an activity j , $j = 0, \dots, n + 1$. Renewable resource types are represented by the set R . Every activity j needs r_{jk} amounts of resource k throughout any period of its time span, in which $r_{0k} = r_n + 1_k = 0$, $k = 1, \dots, r$. All parameters are non-negative integers. The accessibility of every resource category k during all time interval is r_k units, $k = 1, \dots$. A zero parameter value is present in sequential dependencies of the finish-start

type (i.e. E. between the activities, with $FS = 0$. Put another way, if j cannot begin until i has been finished, then activity i comes before j . $G = (V, E)$, in which V represents the collection of activities and E signifies the precedence connections among these activities, is a representation of the structure of a project that uses Activity on Node (AON) networks. For every activity j , where $j=0, \dots, n+1, S_j(P_j)$ is the set of descendants (predecessors). Further assumptions are that $n + 1 \in S_j, j = 0, \dots, n$, and $0 \in P_j, j = 1, \dots, n + 1$. To minimize the project completion time $CT(S) = S_{n+1}$, the goal is to find a schedule S of activity starting times $[s_0, \dots, s_{n+1}]$ that $s_0 = 0$ and the order and resource-limitations are met. Table 2 presents the definition of sets, parameters and assumptions of the problem.

Table 2

Problem description

Item	Description
Sets	
R	The set of renewable resource types
V	The group of activities
E	The collection of sequential dependencies among these activities
$G = (V, E)$	A representation of the structure of a project that uses Activity on Node (AON) networks
P_j	Set of descendants (predecessors)
Parameters	
d_j	Represents the duration of an activity j
r_{jk}	Every activity j needs r_{jk} units of resource k during each period of its duration
F_j	Activity j finish time
S_j	Activity j start time
n	Number of activities
r_k	The supply of each resource category k during every time frame
Assumptions	
0	Project's start activity = Dummy activity
$n+1$	Project's finish activity = Dummy activity
$d_0 = 0$	Project's start activity duration
$d_{n+1} = 0$	Project's finish activity duration
s_0	First activity start time
s_{n+1}	Last activity start time
$FS=0$	A zero parameter value is present in precedence relations of the finish-start type
$n + 1 \in S_j$	$j = 0, \dots, n$
$0 \in P_j$	$j = 1, \dots, n + 1$
$CT(S) = S_{n+1}$	Project completion time which in this case is the last activity start time

4. Methodology

4.1. The proposed Genetic Algorithm

In the 1960s and 1970s, John Holland created and popularized Genetic Algorithms, or GA. The principles of natural selection and natural genetics form the foundation of GA (Damak et al., 2009). It is determined by gathering a number of initial viable solutions that are produced at random or by applying a heuristic known as "the initial population.". Each person in the in question group of individuals is identified as a genetic representation, which provides a clue into how to solve an issue. "Genes" are a set of elements that make up a chromosome. Every problem needs a different encoding according to GA. It also requires a fitness function that gauges the quality of each chromosome that is currently present in the population. With this technique, the reproduction tool creates offspring chromosomes by using a crossover operator to try and select the parents. These progeny possess the ability to randomly undergo a mutation operation through local gene modification, and the elite selection strategy is employed to transfer the most advantageous individuals to the newly formed group. The chromosome elements that result from the creation of the new population are evaluated according to their unique fitness values. Until the termination condition is met, this iterative process is carried out (Fung et al., 2008; Sarkar 2018). According to the figure below, genetic algorithms generally have common steps, which are described below as well as Fig. 1.

- defining the objective function.
- specifying the size of the variables, which is the length of the chromosome string.
- specifying and adjusting GA parameters.
- the machine learning-based initial population is generated.
- For each chromosome, its fitness function is calculated.

- By using the roulette wheel, the fittest chromosomes as input for the next step is selected.
- the single-point crossover operator on the chromosomes is performed.
- the mutation operator on the selected chromosomes is performed.

These steps continue until a certain repetition is reached. Each step of this process will be described in the sections that follow.

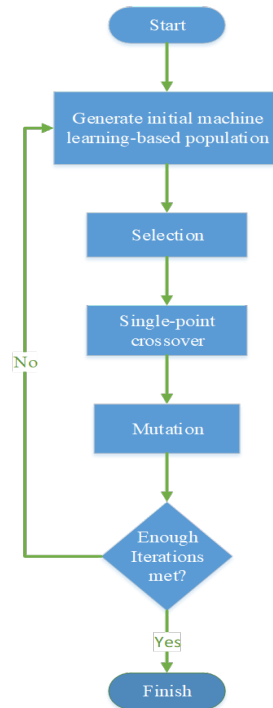


Fig. 1. The flowchart of the research proposed GA

4.1.1. Encoding scheme

There are a total of five encryption schemes available in the literature of this subject, two of which are the best. The activity list (AL) and random-key (RK) schemes are the most effective; they have drawn the greatest attention and use. An activity list scheme gives a feasible precedence activity sequence where each activity is arranged so that it comes prior to all of its following tasks and subsequent all of its preceding tasks. An activity's placement indicates the order of its importance in timeline management. A random-key approach uses position to indicate the activity index and random key to indicate an activity's priority. According to Kolisch and Hartmann (2006), the activity list scheme performs more effectively for the RCPSP compared to the random-key approach. As such, the suggested genetic algorithm incorporates it.

4.1.2. Initial population

The study's starting population can be generated heuristically or at random. According to Lee and El-Sharkawi (2008), the initial population in this study is created at random in order to broaden the search space and find advantageous regions.

4.1.3. Decoding procedure (timetable creation method)

Transferring the encoding scheme toward a feasible schedule is done with a schedule generation scheme (SGS). In essence, the schedule generation scheme grows a local schedule, under the conditions of resource restrictions and order dependencies, gradually into a complete timetable grounded in the regulation of the activity quantity stage finish time. There are two kinds of schedules: parallel and serial. Active schedules are always produced by the activity-oriented serial schedule generation scheme (SSGS). Only the non-delay schedules are created by the time-oriented Parallel Schedule Generation Scheme (PSGS). Among the group of ongoing schedules, of which the collection of schedules without delays merely a portion, the optimal solution is always found. Because of this, the PSGS might overlook the best option (Kolisch 1996). To translate an activity list scheme to a particular activity schedule, the SSGS is used in current study.

4.1.4. Infeasible tackling procedure (Repair strategy)

Since the population generated by the algorithm does not follow the rules of activity predecessors and it is only a permutation of genes (activities), in order to convert the randomly generated schedules (chromosomal strings) into feasible schedules (schedules in which the predecessors of each activity in the chromosome string have been respected) the repair approach has been used. From now on, all individuals who were randomly generated will have a logical sequence of activities and each one will have its own fitting function. One of the remarkable methods in this research is the repair strategy that is able to convert the produced population into feasible ones. After generating the population and converting them into valid chromosomes, we calculate the completion time of each schedule by forward calculation. Since our goal in this issue is to reduce the length of the project completion time, the lower the value of the fitting function is, the better, and it indicates that the project will be completed in a shorter period of time.

4.1.5. Selection

Higher quality individuals are given preference by the selection operator, who uses the fitness data that is available to ensure that their content is carried over towards the subsequent iteration regarding the assessment. This offers an approach to influence this equilibrium toward utilization by emphasizing the superior individuals or toward discovery by offering even the worst individual comparable chances to survive (Lee and El-Sharkawi 2008). The current study aims to strike a balance between exploration and exploitation.

In this study the selection operator is performed by the Roulette wheel. It chooses the chromosomes that have a high value of the fitness function by giving it a more share in the selection phase. Therefore, chromosomes with higher fitness functions have the high chance of being selected and going to the next phase. Subsequently the chance of producing a better population is increased which results in better offspring. For example, we have 100 chromosomes, whose values are between 120 and 165. 54% of these chromosomes have 65 values. Because they have the largest contribution, the probability of their selection is higher than the rest of the chromosomes. As a result, the probability of going to the next generation is very high. As the objective of this research is to shorten the project's finishing time, any chromosome that has a lower fitting function is better. Therefore, chromosomes with lower values are less likely to pass to the next generation. To increase this probability, the genetic algorithm increases the reduction rate of these chromosomes with high values to low values by performing crossover and mutation operators. In addition, in this study, the approach of relocating genes has been used. In this way, before generating the initial population and entering the algorithm, we increase the amount of production of chromosomes that have a lower fitness function by using decision tree analysis. This method will be explained in the subsequent section.

4.1.6. Crossover operator

For every set of parents, the crossover operator is applied, resulting in two offspring who share some traits with their parents. Peteghem and Vanhoucke (2010) conducted an examination of multiple possible crossover functions and confirmed that the single-point crossover provides the optimal efficiency in the RCPSP. Consequently, this research uses the one-point crossover. A random whole number, r , is chosen for the one-point crossover. The parent chromosome contains a copy of all the data from the activity list that is still there. It is possible to recombine genetic fragments of the elite individuals with non-elite ones through the crossover operation, increasing the latter's likelihood of quality. In Fig. 2 an example of the process of a single point crossover operator on a schedule with 10 activities is shown.

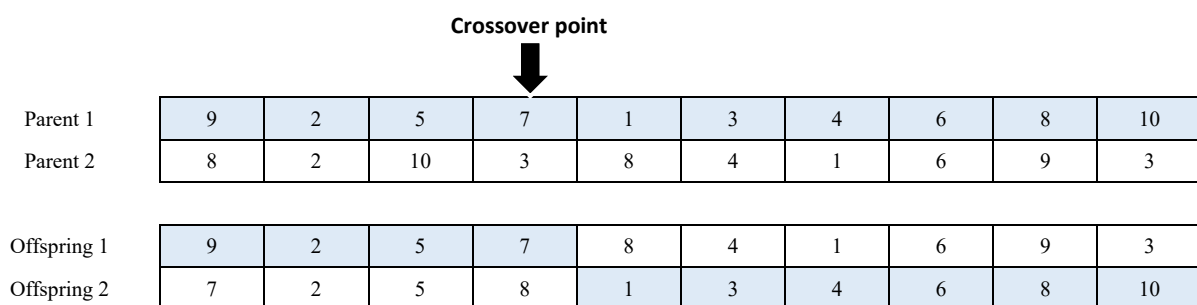


Fig. 2. Single point crossover process on a schedule with 10 activities

4.1.7. Mutation operator

The second lead character in GA is mutation. The solution is altered by mutation operators when they disrupt it. Mutation is a process that involves random alteration. The mutation rate is the indication of its magnitude. To search for the genetic algorithm, the mutation operator adjusts specific genes in a particular individual (liu et al., 2019). In this study, the percentage of mutation was 0.3 and the mutation rate was considered to be 0.02 based on experience. In Fig. 3 an example of the process of the mutation operator on a schedule with 10 activities is shown.

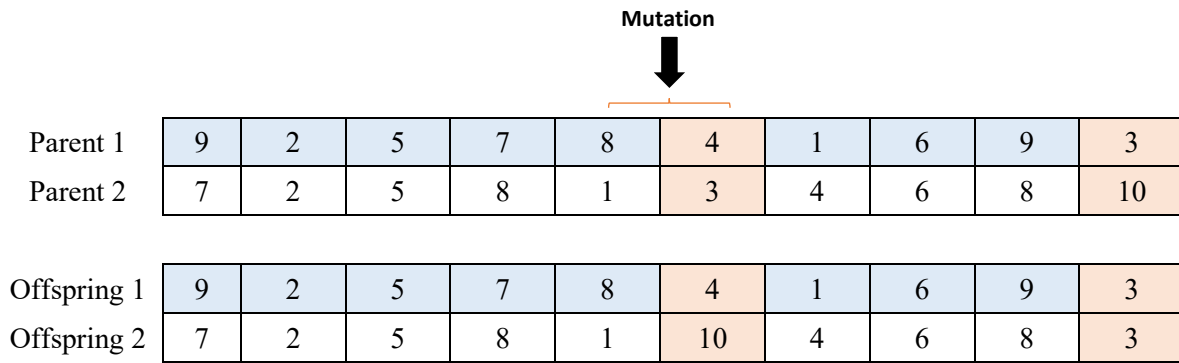


Fig. 3. Mutation process on a schedule with 10 activities

4.2. Data overview

The well-known J60 benchmark set, created with ProGen for the RCPSP, was utilized to demonstrate the efficacy of the suggested ML-GA (available at <https://www.om-db.wi.tum.de/psplib/library.html>). The set includes 60 non-dummy activities across 491 instances. Every time, a single mode that lasts anywhere from one to ten periods can be used to complete an activity. Four different types of renewable resources are included in each instance. The parameters of network and resources were generated and determined by Vanhoucke et al. (2016). Also available at <https://www.projectmanagement.ugent.be/research/data>. Ten networks have been chosen for this study. The output of each network has been used for machine learning analysis, i.e. e. testing and training of model. Every network has its own unique characteristics. The topology of the network and the resource parameters—which in this study are renewable form and single-mode. Table 3 explains the indicators of activity, network and resource.

Table 3

Activity and network and resource indicators

Type	Indicators	Description	Literature
Activity indicators	Number of activities	The count of non-dummy nodes within the network	-
	Network indicators	Coefficient of Network Complexity (CNC)	The overall count of precedence relations (arcs) divided by the overall count of project tasks (nodes) in the network.
	Order Strength (OS)	The total number of precedence relations (immediate and secondary predecessors) compared to the hypothetical upper limit of priority relationships.	Mastor (1970)
	Serial/Parallel (SP)	Evaluating the proximity of a project network to either bare (SP = 1) otherwise pure (SP = 0).	Vanhoucke et al. (2008)
	Activity Distribution (AD)	Project levels and the distribution of activities.	Vanhoucke et al. (2008)
	Length of Arcs (LA)	Each arc has a short length of 1 in terms of the advancing stage of its terminal node and the beginning nod.	Vanhoucke et al. (2008)
	Topological Float (TF)	The topological float of a precedence relation refers to the quantity of level an activity can change without exceeding the upper limit level of the network set by SP.	Vanhoucke et al. (2008)
	Length of arcs I5	The LA is comparable to the measurement, but for extended arcs with a level difference more than 1 between their opening and closing nodes.	Vanhoucke et al. (2008)
Resource indicators	Resource Factor (RF)	The mean quantity regarding resources needed for each activity.	Pascoe (1966)
	Resource Use (RU)	The amount of resource categories required for each activity.	Demeulemeester et al. (2003)
	Resource Strength (RS)	Controversy persists regarding the connection between resource demand and network topology information.	Cooper (1976)
	Resource Constrainedness (RC)	The ratio of a resource's availability to its average demand across all activities.	Patterson (1976)

All the network and resource parameter data for the J60 dataset is calculated and summarized in Table 4.

Table 4

The summarize of j60 data set parameters

	# Act	CNC	OS	SP (or I2)	AD (or I3)	LA (or I4)	I5	TF (or I6)	# RR	RF	RU	RS	RC
Min	60	1.45	0.203	0.119	0.163	0	0.621	0.067	4	0.25	1	0.141	0.068
Max	60	2.083	0.545	0.271	0.51	0.076	0.86	0.499	4	1	4	1	0.491
Ave.	60	1.767	0.357	0.187	0.312	0.025	0.770	0.236	4	0.627	2.509	0.603	0.231
StDev	0	0.259	0.087	0.024	0.068	0.015	0.038	0.072	0	0.280	1.119	0.292	0.098

In section 6, it will be explained how to use them to create the initial population and feed it to the machine learning model. Table 5 provides a summary of the parameters for the ten networks that were used in this study.

Table 5

The summary of 10 selected networks parameters used in this study

Network	# Act	CNC	OS	SP (or I2)	AD (or I3)	LA (or I4)	I5	TF (or I6)	# RR	RF	RU	RS	RC
J609_1	60	1.45	0.227	0.119	0.253	0.021	0.772	0.074	4	0.754	3.017	0.199	0.274
J609_3	60	1.45	0.25	0.186	0.295	0.003	0.723	0.286	4	0.754	3.017	0.187	0.34
J609_6	60	1.45	0.271	0.186	0.341	0.006	0.723	0.313	4	0.754	3.017	0.2	0.287
J609_7	60	1.45	0.237	0.169	0.408	0.011	0.741	0.337	4	0.754	3.017	0.196	0.303
J6013_1	60	1.45	0.246	0.169	0.408	0.005	0.73	0.32	4	1	4	0.201	0.271
J6013_3	60	1.45	0.227	0.169	0.398	0.011	0.785	0.32	4	1	4	0.199	0.223
J6013_5	60	1.45	0.215	0.153	0.4	0.014	0.813	0.284	4	1	4	0.198	0.25
J6013_9	60	1.45	0.268	0.203	0.385	0.014	0.739	0.358	4	1	4	0.204	0.242
J6041_3	60	2.083	0.429	0.186	0.295	0.048	0.787	0.189	4	0.754	3.017	0.197	0.318
J6045_1	60	2.083	0.428	0.186	0.273	0.038	0.785	0.21	4	1	4	0.204	0.26

The following is the structure of the matrix known as the data matrix, which is provided to the machine learning model: The rows represent the records, which are our networks and are referred to as instances. The problem parameters (network), the solution and the objective function are the headings for the three categories that make up the columns. The time it required to finalize a project (C_{max} of the network) is the fitness function (objective function) of each chromosome string. Every schedule has a completion time and is generated by the genetic algorithm. Every network is known as a project. The problem's parameters and schedules are considered as input variables, while the project completion time is considered an output variable. Every item is displayed in Table 6.

Table 6

The data matrix elements

	Problem Parameters					The Solution					The objective Function
	N0	P_1	P_2	...	P_m	X_1	X_2	...	X_n	X_n	Z
Instances	1										
	2										
	3										
	·										
	·										
N											
	Input Variables										Output Variables

4.3. The proposed learnheuristic method

Classification, as a supervised ML approach, proves to be highly proficient in the realm of predictive data analytics. This technique relies on the assimilation of knowledge from historical or training data to facilitate the allocation of novel input instances (e.g., project cases) into distinct output classes of dependent variables based on pertinent values of independent variables. The primary focus of the present research is on the adoption of the classification methodology due to its adeptness in managing intricate interrelations among variables and its efficacy in addressing categorical variables. This study utilized the decision tree (DT) algorithm as one of its classification algorithms. The selection of ML classification algorithms was made by analyzing the literature and recognizing their suitability for large data sets with proven, satisfactory results. The network indicators and resource indicators and chromosome strings are considered as input of the machine learning model. This is called a data matrix. This data is given to the model for learning. 80% of the input is considered as training and 20% as testing. Its steps are as follows:

- Generate initial population

- o Generating an initial feasible population before algorithm iteration process
- o Apply the infeasible tackling procedure to infeasible solutions
- o Apply forward-backward iteration to solutions
- o Evaluate each individual
 - Data matrix establishment
- o Establish set of features with inputs and output variables
- o Chromosome strings collection
- o Project indicators collection
 - ML model training
- o Training set
 - § Model parameters
 - § Model training
- o Testing set
 - § Predicting by the trained model
 - Getting classification model
 - GA algorithm
- o Coding relocation procedure
 - o Generate initial population
 - o Roulette wheel selection
 - o Single-point crossover
 - o Mutation
 - o Finish

So far, the steps of implementing the algorithm and describing and explaining the problem and the purpose of the research have been stated. In this stage, all of the steps involved in this procedure will be described below as well as Fig. 4.

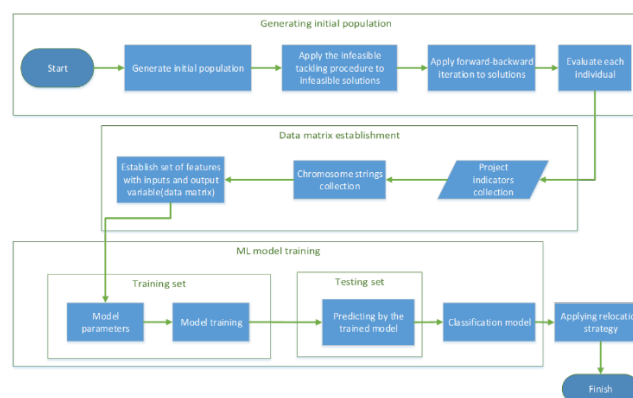


Fig. 4. The diagram of the recommended machine learning procedure

4.3.1. Generating initial feasible population

Now all the calculation processes from building the model, generating the random solution, converting it into a feasible solution and calculating the objective function (project completion time) were explained. In the next step, the model should have connected to the algorithm so that it has become possible to generate a large population of feasible schedules. In this case, they can be considered as input to the machine learning model. Considering the time of different runs done by the algorithm to address the challenge, 20 iterations to solve the model by the algorithm seemed reasonable.

1	2	3	6	7	10	43	16	21	29	17	28	48	25	11
44	9	12	18	37	26	30	55	23	35	13	41	46	4	8
34	39	15	57	59	19	53	24	27	38	31	5	14	36	42
22	32	45	40	50	20	49	54	58	52	33	47	51	41	56
60	62													

Fig. 5. An example of a feasible schedule with 62 activities used in this study

According to the experience in using meta-heuristic algorithms in solving NP problems -hard, the crossover rate considered 0.8, the mutation percentage considered 0.3, and the mutation rate considered 0.02. Single-point crossover has been used to perform crossover and roulette wheel has been used to select parents in each generation. The number of population generated from each project network is considered to be 1000 in order to have a large variety of schedules. As a result, the crossover operation will be applied to 800 populations and the mutation operation will be applied to 300 populations. Due to the high volume of data and their analysis and Getting the desired output, machine learning and decision tree tools are used in this section. Now having finished the algorithm's preparation and model integration, the next step is to generate the initial population. According to the strategy of this research, which is the manipulation of the initial population and to move the arrangement of activities before entering the algorithm The desired population is created before entering the iteration process so that none of the genetic operators are performed on it, but only randomly feasible schedules are generated that each of them has a fitness function which is the completion time of the project.

4.3.2. Data matrix establishment

The initial population from each 10 selected networks which were mentioned in the research data collection section imported from MATLAB to Excel. It should be noted that only the schedules (chromosomes) and the completion time (C_{max}) of each schedule must be entered into Excel, and nothing should be done with other parameters from the output of the problem. 1000 schedules from each network are produced, which makes a total of 10000 schedules. These schedules are along with 14 other parameters that were described in the data overview section. Table 7 shows the data matrix format (columns and rows along with schedules) for all 10 networks used in this research.

Table 7
The data matrix of 10 selected networks in this study

Name	#Act	CNC	...	RC	q ₁	q ₂	...	q ₆₁	q ₆₂	C _{max}
J609_1	60	1.45	...	0.274	1	2	...	59	62	99
J609_3	60	1.45	...	0.34	1	2	...	59	62	117
J609_6	60	1.45	...	0.287	1	4	...	60	62	131
J609_7	60	1.45	...	0.303	1	4	...	61	62	127
J6013_1	60	1.45	...	0.271	1	2	...	60	62	131
J6013_3	60	1.45	...	0.223	1	4	...	61	62	102
J6013_5	60	1.45	...	0.25	1	3	...	59	62	116
J6013_9	60	1.45	...	0.242	1	3	...	61	62	118
J6041_3	60	2.083	...	0.318	1	4	...	61	62	119
J6045_1	60	2.083	...	0.26	1	3	...	60	62	114

5. Computational experiments

5.1. Machine learning model training

At this point the data needs to be imported into the SPSS Modeler neural network. Since the main objective of the work is to lower C_{max} , that is considered as the target of the input data. This means that all other parameters are considered as inputs and only C_{max} as output. Table 8 shows the inputs and output of the model.

Table 8
The inputs and Target (output) of the neural network model

input	input	input	input	input	input	input	input	input	Target
CNC	OS	SP	...	q ₁	q ₂	...	q ₆₂	C _{max}	

The decision tree tool analyzes and extracts rules from the data matrix. In the partition node of the SPSS software, 80% of the data was considered for model training and 20% for model testing. SPSS software allows us to display our desired output using one of three types of decision trees. Compared to the other two decision trees, the CRT decision tree node gives us a more compact output, which is why we chose it. In fact, fewer rules are extracted this way making them more suitable for understanding and implementing. Other decision trees are CHAID and C5.0. The output of the decision tree after neural network model testing and training is shown in Fig. 7.

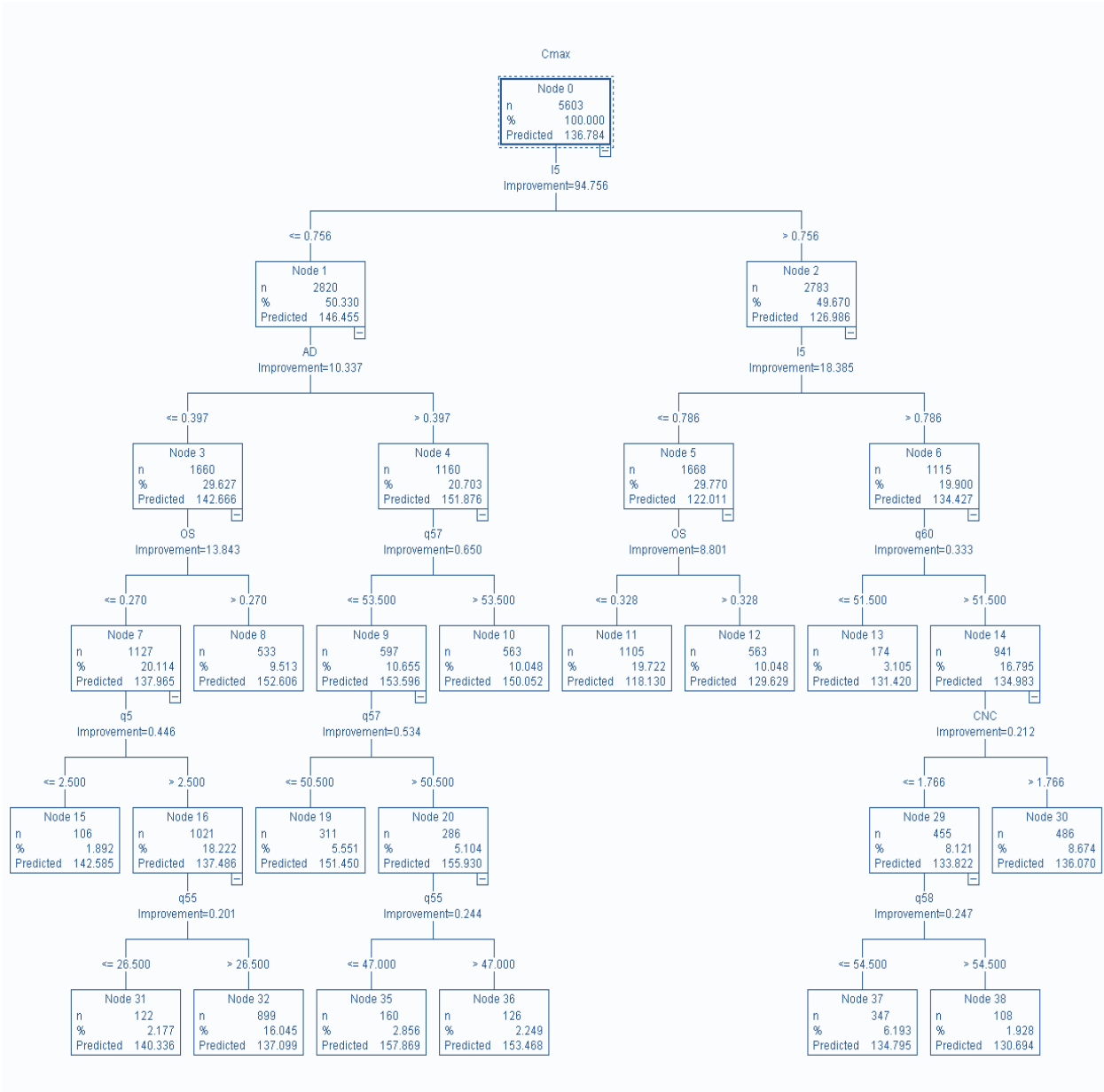


Fig. 6. The extracted decision tree after training the model

Based on Fig. 6, a branch with a lower value is selected. The lowest point of the tree, which represents the value of the predicted solution for each branch, is indicated by the word “predicted” in the leaf. The value of the selected leaf is expected to be 130,094 (in the lower right corner of Fig. 6). To apply the rules, this branch is used as the final solution of the decision tree. The data matrix in the Excel file is first used to determine which network or networks contain these rules. The guidelines that cause the project’s to be reduced according to the prediction of the decision tree. Second, these guidelines should be followed when creating the initial population. This allows a population of schedules with the required features to be achieved. This is achieved by rearranging the order of activities. In other words, by shifting the position of the activity. This procedure is called displacement procedure which is explained in further paragraphs. The branch selected to create the intended initial population and apply the current rules therein is described for clarity. The extracted rules and branches opened from root to leaf are shown in Fig. 7, which is an alternative representation of the decision tree output. It should be noted that when a number is placed between two parentheses it shows the gene that has a certain value but if it does not have parentheses, it shows the place of that gene. For example, q58 means the 58th gene of the chromosome string and q(58) means the gene that has the value of 58 in the chromosome string. If a particular gene has a value of 30 (q(30)) in the 10th place of a chromosome string it is represented q10=30.

```

I5 <= 0.756 [ Ave: 146.455, Effect: 9.67 ]
  AD <= 0.397 [ Ave: 142.666, Effect: -3.788 ]
    OS <= 0.270 [ Ave: 137.965, Effect: -4.701 ]
      q5 <= 2.500 [ Ave: 142.585, Effect: 4.62 ] => 142.585
      q5 > 2.500 [ Ave: 137.486, Effect: -0.48 ]
        q55 <= 26.500 [ Ave: 140.336, Effect: 2.85 ] => 140.336
        q55 > 26.500 [ Ave: 137.099, Effect: -0.387 ] => 137.099
      OS > 0.270 [ Ave: 152.606, Effect: 9.94 ] => 152.606
    AD > 0.397 [ Ave: 151.876, Effect: 5.421 ]
      q57 <= 53.500 [ Ave: 153.596, Effect: 1.72 ]
        q57 <= 50.500 [ Ave: 151.45, Effect: -2.146 ] => 151.45
        q57 > 50.500 [ Ave: 155.93, Effect: 2.334 ]
          q55 <= 47 [ Ave: 157.869, Effect: 1.939 ] => 157.869
          q55 > 47 [ Ave: 153.468, Effect: -2.462 ] => 153.468
        q57 > 53.500 [ Ave: 150.052, Effect: -1.824 ] => 150.052
I5 > 0.756 [ Ave: 126.986, Effect: -9.799 ]
  I5 <= 0.786 [ Ave: 122.011, Effect: -4.974 ]
    OS <= 0.328 [ Ave: 118.13, Effect: -3.881 ] => 118.13
    OS > 0.328 [ Ave: 129.629, Effect: 7.617 ] => 129.629
  I5 > 0.786 [ Ave: 134.427, Effect: 7.441 ]
    q60 <= 51.500 [ Ave: 131.42, Effect: -3.007 ] => 131.42
    q60 > 51.500 [ Ave: 134.983, Effect: 0.556 ]
      CNC <= 1.766 [ Ave: 133.822, Effect: -1.161 ]
        q58 <= 54.500 [ Ave: 134.795, Effect: 0.973 ] => 134.795
        q58 > 54.500 [ Ave: 130.694, Effect: -3.128 ] => 130.694
      CNC > 1.766 [ Ave: 136.07, Effect: 1.087 ] => 136.07
    
```

Fig. 7. The extracted rules from decision tree

In Table 9 the selected highlighted underlined rules in Fig. 8 are given.

The rules extracted using the decision tree are written as code in MATLAB. To do this, the concept of neighborhood is used. There are three methods to create it:

- Insertion
- Swap
- Reversion

In this study, because each produced chromosome is a feasible schedule, if the concepts of swap and reversion are used, the arrangement of genes (activities) which is based on the predecessor of each activity will be greatly disturbed and most probably no change will be observed in the completion time of each chromosome. For this reason, the insertion approach is used. In this approach, by replacing only one gene, it is possible to improve the completion time of each chromosome (schedule). In Fig. 8 the procedure is shown:

1	2	3	6	7	10	43	16	21	29	17	28	48	25	11
44	9	12	18	37	26	30	55	23	35	13	41	46	4	8
34	39	15	57	59	19	53	24	27	38	31	5	14	36	42
22	32	45	40	50	20	49	54	58	52	33	47	51	41	56
60	62													

Fig. 8. An example of insertion procedure on a real schedule with 62 activities

	1	2	3	6	7	10	43	16	21	29	17	28	48	25
11	44	9	12	18	37	26	30	23	35	13	41	46	4	8
34	39	15	57	59	19	53	24	27	38	31	5	14	36	42
22	32	45	40	50	20	49	54	58	52	33	47	55	51	41
56	60	62												

Fig. 9. After the insertion

As shown in Fig. 9, q(55) is placed at q58 and the chromosome string shifts one unit to the right. According to the decision tree, this displacement leads to a decrease in C_{max} in a number of schedules (not all of them).

5.2. The role of GA algorithm

After extracting the relevant rules, it is time to apply them in MATLAB software. It should be noted that among the 10 networks given to the software for analysis, the j6013_5 network was chosen as the selected network to apply the extracted rules. The next part shows the impact of the new approach of this research in the generation of the initial population (the population on which machine learning was employed and the rules was applied) compared to the normal approach (which is the population generated without the application of machine learning and applying the rules). In Table 10, the q60 gene corresponds to the number 52. Out of 10,000 chromosomes, 336 have q60=52. But the number of q60=52 after applying our approach, which represents the manipulation of the initial population through the application of extracted rules, is 420, which corresponds to a difference of 84, and this shows the impact of machine learning on the population generation and code, which was written to change the genes in the chromosomes. The situation is similar with the q58 gene; whose number is 23 before applying our approach. After applying our approach that q58 = 55, it reached 105, which also indicates the impact of using machine learning to change the initial population. Table 10 shows up to 5 repetitions of generation of the initial population using GA (normal approach) and the learnheuristic (ML-GA). The difference between the two approaches is shown in the difference (last) column.

Table 10
Number of schedules with q60 > 51.5 for both methods

Network Name		ML-GA	GA	Difference
J6013_5	q60	418	323	95
	q60	60	23	37
	q60	420	336	84
	q60	27	39	-12
	q60	114	112	2
	Sum	1039	833	206

Fig. 10 shows the difference between GA and the proposed learnheuristic method (ML-GA) approaches from the perspective of generating the desired population after applying the rules extracted from the decision tree.

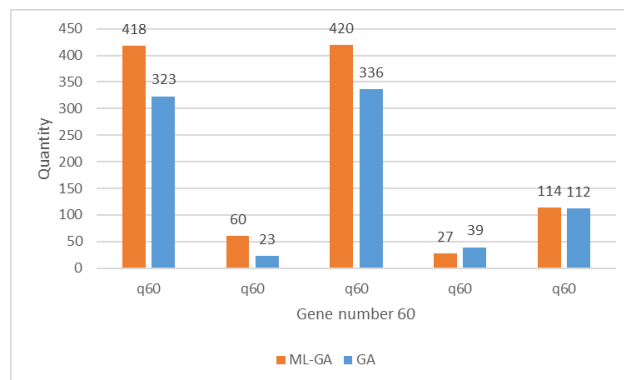


Fig. 10. Comparison of two methods from the point of view of the number of schedules produced with q 60 > 51.5

6. Results

The generated population was given to the algorithm to solve the problem in different iterations (20 iterations). The NFE index was used as a measure of efficiency. NFE represents the number of evaluation function calls in each iteration. Since

converting the transformation of the chromosome into a feasible schedule is defined as a function in MATLAB, NFE basically records the number of calls to the function to get the solution. The algorithm was run five times in GA and ML-GA approaches and its findings are demonstrated in Table 11.

Table 11
The comparison between normal GA and learnheuristic (ML-GA) approaches

Name	Ac	CN	OS	SP	A	LA	I5	TF	R	R	R	RS	RC	Iterati	C_{max}	NFE(G)	NFE	%
J6013	60	1.4	0.21	0.15	0.	0.01	0.81	0.28	4	1	4	0.19	0.2	14	113	20800	1640	21.15
J6013	60	1.4	0.21	0.15	0.	0.01	0.81	0.28	4	1	4	0.19	0.2	9	112	23000	1090	52.61
J6013	60	1.4	0.21	0.15	0.	0.01	0.81	0.28	4	1	4	0.19	0.2	20	111	23000	2300	0.00
J6013	60	1.4	0.21	0.15	0.	0.01	0.81	0.28	4	1	4	0.19	0.2	14	113	17500	1640	6.29
J6013	60	1.4	0.21	0.15	0.	0.01	0.81	0.28	4	1	4	0.19	0.2	14	114	23000	1640	28.70
Average=														14	112.	21460	1662	21.75

Fig. 11 shows the percentage of improvement in the comparison of GA and learnheuristic (ML-GA) approaches. In the first run, the GA approach reached the desired solution with 20,800 calls of the evaluation function. In the ML-GA approach, the desired answer was reached with 16,400 calls of the evaluation function. By comparing these two, we have 21.15% improvement in the frequency with which the evaluation function is invoked. Likewise, we have reached the answer in less time. The second run has the highest amount of improvement compared to the others. The GA approach has reached the desired answer with 23,000 calls of the evaluation function. The ML-GA approach has reached the desired answer with 10,900 calls of the evaluation function. By comparing these two, we have 52.61% improvement in the count of time the evaluation function is called. Likewise, we reached the answer more quickly. The next runs are also shown in Fig. 12.

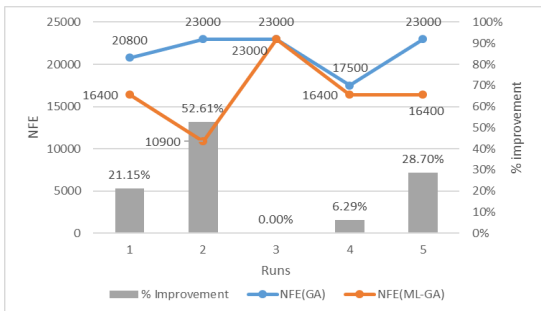


Fig. 11. The percentage of improvement in terms of the number of times the evaluation function is called in both methods

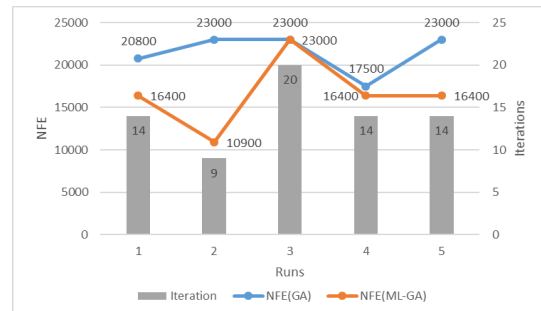


Fig. 12. Comparison of each method in terms of NFE and number of iterations

In Fig. 12, NFE and the iteration count for each execution is evaluated against to reach a better solution. The GA approach in the first run has been solved in iteration 14 with 20,800 calls of the evaluation function, while the learnheuristic (ML-GA) approach in the first run has been solved in iteration 14 with 16,400 calls of the evaluation function. According to the figure, the second run with the least repetition has reached the desired solution. The GA approach in the first run, in iteration 9 with 20,800 calls of the evaluation function, has been solved, while the ML-GA approach in the first run, in the 14th iteration, has been solved with 10,900 calls of the evaluation function. The next runs are also shown in the figure.

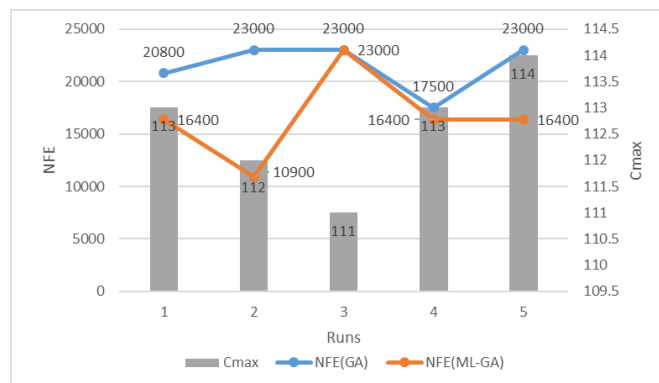


Fig. 13. Comparison of each method in terms of NFE and C_{max}

In Fig. 13, the GA approach with 20,800 calls of the evaluation function has reached the completion time of 113, while the ML-GA approach has reached the completion time of 113 with 16,400 calls. According to the figure, the second run has the biggest difference in the number of calls of the evaluation function to reach the solution.

As it can be seen, C_{max} , which is the project completion time, has an average of 112.6. The improvement rate of each run compared to the traditional GA method is shown in the last column of Table 10 the average of which is 21.75. The strength of the learnheuristic method compared to the classic GA is the number of NFE function calls. The lower the NFE value, the less search effort and time required for the algorithm to reach a better solution in the desired iteration. However, this results in a decrease in the project's cost, particularly in cases where repetitive work is involved. This shows the efficiency of the ML-GA method compared to GA by 21.75%.

7. Conclusion

This paper proposed the use of a learnheuristic method to solve the RCPSp combined of GA and machine learning technique. Activity Lists (AL) were utilized for encoding, and SSGS was taken into consideration for decoding. A repair strategy was employed in the procedure for handling infeasibility to convert unfeasible schedules into valid individuals. To form a learnheuristic, The GA was enhanced through the use of machine learning. In order to achieve this, the generated schedules were analyzed using decision tree prior to the algorithm. Next, the chosen network was subjected to the rules that were extracted from the decision tree output. Subsequently, by giving this machine learning-based population to algorithm, it reached the desired solution by continuing to run until a certain number of iterations. In PSPLIB, it was evaluated to resolve the popular standard set J60. When the outcomes were contrasted with the testing results. of the standard GA, the efficacy and resilience of the proposed method in resolving these issues were demonstrated. Additionally, the results indicated a 21.75 percent improvement. Future research can expand on the suggested model in the following directions:

- Measuring activity duration and other resource concepts with fuzzy sets and comparing the results.
- Exploiting other metaheuristic algorithms with machine learning concepts such as swarm optimization to improve the result of the problem and extend the literature.
- Extending the suggested technique to another machine learning approaches such as reinforcement learning and deep learning.
- Resolving these kinds of issues by combining machine learning with other RCPSp extensions such as MRCPSP
- Development of concepts and techniques used in this research for project cost supervision

References

- Afshar, M. R., Shahhosseini, V., & Sebt, M. H. (2022). A genetic algorithm with a new local search method for solving the multimode resource-constrained project scheduling problem. *International Journal of Construction Management*, 22(3), 357-365. <https://doi.org/10.1080/15623599.2019.1623992>
- Bettemir, Ö. H., & Sonmez, R. (2015). Hybrid genetic algorithm with simulated annealing for resource-constrained project scheduling. *Journal of Management in Engineering*, 31(5), 04014082. [https://doi.org/10.1061/\(ASCE\)ME.1943-5479.00003](https://doi.org/10.1061/(ASCE)ME.1943-5479.00003)
- Blazewicz, J., Lenstra, J. K., & Kan, A. R. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete applied mathematics*, 5(1), 11-24. [https://doi.org/10.1016/0166-218X\(83\)90012-4](https://doi.org/10.1016/0166-218X(83)90012-4)
- Cheng, M. Y., Tran, D. H., & Wu, Y. W. (2014). Using a fuzzy clustering chaotic-based differential evolution with serial method to solve resource-constrained project scheduling problems. *Automation in Construction*, 37, 88-97. <https://doi.org/10.1016/j.autcon.2013.10.002>
- Cooper, D. F. (1976). Heuristics for scheduling resource-constrained projects: An experimental investigation. *Management Science*, 22(11), 1186-1194. <https://doi.org/10.1287/mnsc.22.11.1186>
- Damak, N., Jarboui, B., Siarry, P., & Loukil, T. (2009). Differential evolution for solving multi-mode resource-constrained project scheduling problems. *Computers & Operations Research*, 36(9), 2653-2659. <https://doi.org/10.1016/j.cor.2008.11.010>
- Demeulemeester, E., Vanhoucke, M., & Herroelen, W. (2003). RanGen: A random network generator for activity-on-the-node networks. *Journal of scheduling*, 6, 17-38. <https://doi.org/10.1023/A:1022283403119>
- Etmianiesfahani, A., Gu, H., Naeni, L. M., & Salehipour, A. (2022). A forward-backward relax-and-solve algorithm for the resource-constrained project scheduling problem. *SN Computer Science*, 4(2), 104. <https://doi.org/10.1007/s42979-022-01487-1>
- Fung, I. W., Wong, C. K., Tam, C. M., & Tong, T. K. (2008). Optimizing material hoisting operations and storage cells in single multi-storey tower block construction by genetic algorithm. *International Journal of Construction Management*, 8(2), 53-64. <https://doi.org/10.1080/15623599.2008.10773115>
- Golab, A., Gooya, E., Falou, A., & Cabon, M. (2023). A convolutional neural network for the resource-constrained project scheduling problem (RCPSp): A new approach. *Decision Science Letters*, 12(2), 225-238.
- Guo, W., Vanhoucke, M., Coelho, J., & Luo, J. (2021). Automatic detection of the best performing priority rule for the resource-constrained project scheduling problem. *Expert systems with applications*, 167, 114116. <https://doi.org/10.1016/j.eswa.2020.114116>
- Habibi, F., Barzinpour, F., & Sadjadi, S. (2018). Resource-constrained project scheduling problem: review of past and recent developments. *Journal of project management*, 3(2), 55-88. <https://doi.org/10.5267/J.JPM.2018.1.005>

- Hartmann, S., & Briskorn, D. (2022). An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of operational research*, 297(1), 1-14. <https://doi.org/10.1016/j.ejor.2021.05.004>
- Hua, Z., Liu, Z., Yang, L., & Yang, L. (2022). Improved genetic algorithm based on time windows decomposition for solving resource-constrained project scheduling problem. *Automation in Construction*, 142, 104503. <https://doi.org/10.1016/j.autcon.2022.104503>
- Khajesaedi, S., Sadjadi, S., Barzinpour, F & Tavakkoli-Moghaddam, R. (2025). Resource-constrained project scheduling problem: Review of recent developments. *Journal of Project Management*, 10(1), 1-26. DOI: [10.5267/j.jpm.2024.12.002](https://doi.org/10.5267/j.jpm.2024.12.002)
- Kolisch, R. (1996). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 90(2), 320-333. [https://doi.org/10.1016/0377-2217\(95\)00357-6](https://doi.org/10.1016/0377-2217(95)00357-6)
- Kolisch, R., & Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European journal of operational research*, 174(1), 23-37. <https://doi.org/10.1016/j.ejor.2005.01.065>
- Kolisch, R., & Padman, R. (2001). An integrated survey of deterministic project scheduling. *Omega*, 29(3), 249-272. [https://doi.org/10.1016/S0305-0483\(00\)00046-3](https://doi.org/10.1016/S0305-0483(00)00046-3)
- Lee, K. Y., & El-Sharkawi, M. A. (Eds.). (2008). *Modern heuristic optimization techniques: theory and applications to power systems*. John Wiley & Sons.
- Li, C., Wang, F & Chung, T. (2024). Multi-mode multi-skill resource-constrained project scheduling problem with differentiated professional capabilities. *Journal of Project Management*, 9(1), 27-44. DOI: [10.5267/j.jpm.2023.9.002](https://doi.org/10.5267/j.jpm.2023.9.002)
- Liu, Q., Liu, X., Wu, J., & Li, Y. (2019). An improved NSGA-III algorithm using genetic K-means clustering algorithm. *Ieee Access*, 7, 185239-185249. <https://doi.org/10.1109/ACCESS.2019.2960531>
- Liu, Y., Jin, S., Zhou, J., & Hu, Q. (2023). A branch-and-bound algorithm for the unit-capacity resource constrained project scheduling problem with transfer times. *Computers & Operations Research*, 151, 106097. <https://doi.org/10.1016/j.cor.2022.106097>
- Mastor, A. A. (1970). An experimental investigation and comparative evaluation of production line balancing techniques. *Management science*, 16(11), 728-746. <https://doi.org/10.1287/mnsc.16.11.728>
- Pascoe, T. L. (1966). Allocation of resources CPM. *Revue Francaise de Recherche Operationnelle*, 10(38), 31-31.
- Patterson, J. H. (1976). Project scheduling: The effects of problem structure on heuristic performance. *Naval Research Logistics Quarterly*, 23(1), 95-123.6 <https://doi.org/10.1002/nav.3800230110>
- Pedregosa, F. (2011). Scikit-learn: Machine learning in python Fabian. *Journal of machine learning research*, 12, 2825.
- Quinlan, J. R. (1993, June). Combining instance-based and model-based learning. In *Proceedings of the tenth international conference on machine learning* (pp. 236-243).
- Roy, B., & Sen, A. K. (2024). A novel swarm intelligence approach for the constrained scheduling problem. *International Journal of Management Science and Engineering Management*, 19(3), 199-211. <https://doi.org/10.1080/17509653.2023.2248058>
- Sarkar, D. (2018). Hybrid approach for resource optimization and management of bridge projects using bootstrap technique and genetic algorithm. *International Journal of Construction Management*, 18(3), 207-220. <https://doi.org/10.1080/15623599.2017.1315526>
- Sarker, I. H., Furhad, M. H., & Nowrozy, R. (2021). Ai-driven cybersecurity: an overview, security intelligence modeling and research directions. *SN Computer Science*, 2(3), 173. <https://doi.org/10.1007/s42979-021-00557-0>
- Shuvo, O., Golder, S., & Islam, M. R. (2023). A hybrid metaheuristic method for solving resource constrained project scheduling problem. *Evolutionary Intelligence*, 16(2), 519-537. <https://doi.org/10.1007/s12065-021-00675-x>
- Van Peteghem, V., & Vanhoucke, M. (2010). A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 201(2), 409-418. <https://doi.org/10.1016/j.ejor.2009.03.034>
- Vanhoucke, M., Coelho, J., & Batselier, J. (2016). An overview of project data for integrated project management and control. *The Journal of Modern Project Management*, 3(3), 158-158.
- Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B., & Tavares, L. V. (2008). An evaluation of the adequacy of project network generators with systematically sampled networks. *European journal of operational research*, 187(2), 511-524. <https://doi.org/10.1016/j.ejor.2007.03.032>
- Xie, L. L., Chen, Y., Wu, S., Chang, R. D., & Han, Y. (2024). Knowledge extraction for solving resource-constrained project scheduling problem through decision tree. *Engineering, Construction and Architectural Management*, 31(7), 2852-2877. <https://doi.org/10.1108/ECAM-04-2022-0345>



© 2025 by the authors; licensee Growing Science, Canada. This is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (<http://creativecommons.org/licenses/by/4.0/>).