Optimized Resource-Constrained Method for Project Schedule Compression

Moaaz Aly Elkabalawy

A Thesis

In the Department

of

Building, Civil and Environmental Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Civil Engineering)

Concordia University

Montreal, Quebec, Canada

October 2020

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This statement is to certify that the thesis prepared

By: Moaaz Aly Elkabalawy

Entitled: Optimized Resource-Constrained Method For Project Schedule Compression

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Civil Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final Examining Committee:

_____Chair

Dr. S. H. Han

_____Examiner

Dr. R. Dziedzic

_____Examiner

Dr. M. Y. Chen

_____Supervisor

Dr. O. Moselhi

Approved by_____

Dr. Michelle Nokken, Graduate Program Director

October 28, 2020 _____

Dr. Amir Asif, Dean, Gina Cody School of Engineering and Computer Science

ABSTRACT

Optimized Resource-Constrained Method for Project Schedule Compression

Moaaz Elkabalawy

Construction projects are unique and can be executed in an accelerated manner to meet market conditions. Accordingly, contractors need to compress project durations to meet client changing needs and related contractual obligations and recover from delays experienced during project execution. This acceleration requires resource planning techniques such as resource leveling and allocation. Various optimization methods have been proposed for the resource-constrained schedule compression and resource allocation and leveling individually. However, in real-world construction projects, contractors need to consider these aspects concurrently.

For this purpose, this study proposes an integrated method that allows for joint consideration of the above two aspects. The method aims to optimize project duration and costs through the resources and cost of the execution modes assigned to project activities. It accounts for project cost and resource-leveling based on costs and resources imbedded in these modes of execution. The method's objective is to minimize the project duration and cost, including direct cost, indirect cost, and delay penalty, and strike a balance between the cost of acquiring and releasing resources on the one hand and the cost of activity splitting on the other hand.

The novelty of the proposed method lies in its capacity to consider resource planning and project scheduling under uncertainty simultaneously while accounting for activity splitting. The proposed method utilizes the fuzzy set theory (FSs) for modeling uncertainty associated with the duration and cost of project activities and genetic algorithm (GA) for scheduling optimization. The method has five main modules that support two different optimization methods: modeling uncertainty and defuzzification module; scheduling module; cost calculations module; sensitivity

analysis module; and decision-support module. The two optimization methods use the genetic algorithm as an optimization engine to find a set of non-dominated solutions. One optimization method uses the elitist non-dominated sorting genetic algorithm (NSGA-II), while the other uses a dynamic weighted optimization genetic algorithm. The developed scheduling and optimization method is coded in python as a stand-alone automated computerized tool to facilitate the needed iterative rescheduling of project activities and project schedule optimization.

The developed method is applied to a numerical example to demonstrate its use and to illustrate its capabilities. Since the adopted numerical example is not a resource-constrained optimization example, the proposed optimization methods are validated through a multi-layered comparative analysis that involves performance evaluation, statistical comparisons, and performance stability evaluation. The performance evaluation results demonstrated the superiority of the NSGA-II against the dynamic weighted optimization genetic algorithm in finding better solutions. Moreover, statistical comparisons, which considered solutions' mean, and best values, revealed that both optimization methods could solve the multi-objective time-cost optimization problem. However, the solutions' range values indicated that the NSGA-II was better in exploring the search space before converging to a global optimum; NSGA-II had a trade-off between exploration (exploring the new search space) and exploitation (using already detected points to search the optimum). Finally, the coefficient of variation test revealed that the NSGA-II performance was more stable than that of the dynamic weighted optimization genetic algorithm.

It is expected that the developed method can assist contractors in preparation for efficient schedule compression, which optimizes schedule and ensures efficient utilization of their resources.

ACKNOWLEDGMENT

First and foremost, I’m very grateful to God for granting me showers of blessings throughout my research, and for helping me complete the research successfully.

I would like to express my special thanks of gratitude to my research supervisor, Professor Osama Moselhi, for his patience, motivation, enthusiasm, and immense knowledge. His invaluable guidance helped me in all time of research and writing of this thesis. I could not have imagined a better advisor and mentor for my master's study. His dynamism, sincerity, vision, and motivation have deeply inspired me. It was a great privilege and honor to work and study under his guidance.

Finally, I would like to thank my parents for their love, prayers, caring and sacrifices for educating and preparing me for my future. Also, I express my thanks to my brothers and friends for their endless support and valuable prayers.

DEDICATION

This thesis is dedicated with love, admiration, and respect

To my dear father Prof. Aly Elkabalawy,

my kind mother Sahar Elgendy,

and my beloved Brothers Eng. Mohammed and Dr. Abdalla;

Table of Contents

List of Tables

List of Figures

Notations

N	Number of activities
NC	Number of critical activities
NN	Number of non-critical activities
M_j	Number of modes for executing activity j; j=1,2,…….N
	under mode m and zero otherwise
P	Number of resource types required by the project
D_{jm}	Duration of activity j running in mode m; m=1,2…. M_j
	time t; t= ES_j to LF_j
T=	Actual project duration
PC	Project cost
F	Contract project duration
B	Delay penalty/ Bonus payment
CA_p	Acquiring cost of resource p
CR_p	Releasing cost of resource p

CS_j	Splitting cost of activity j
IC	Indirect cost
C_{jm}	Direct cost of activity j under mode m
x_{jm}	Binary variable equals to 1 when activity j is performed under mode m and zero otherwise
y_{tj}	Binary variable equals to one when activity j is progress at time t; t= $ES_j \ to \ LF_j$
z_{jt}	Required resources by activity j during time t.
NL_j	Number of times activity j is split.
D_{tp}	Number of resources p released during time t
I_{tp}	Number of resources p acquired during time t
S_j	Start time of activity j; j=1…,nc
F_j	Finish time of activity j; j=1…,nc

1 CHAPTER ONE: INTRODUCTION

1.1 Background

Delays and cost overruns are routine incidents that occur at many construction projects, despite the project managers' considerable efforts to execute projects within the targeted budgets and dates. According to a study by Baloi and Price (2003), a sizable majority (63%) of 1778 construction projects funded by the World Bank exceeded their budgets. This problem is more severe in developing countries where time and cost overruns frequently exceed 100% of the projects' planned cost and duration (Memon, Rahman and Abdullah, 2010). Therefore, delays and cost overruns can be possible reasons behind inevitable disputes between contractors and clients.

Accordingly, schedule compression has been introduced in the industry and studied intensely in the literature. It may be in the interest of both clients and contractors to reduce the project schedule for various reasons. Contractors tend to compress project durations to meet contractual obligations, recover from delays experienced during project execution, and determine the least project's duration. On the contrary, clients tend to enforce accelerated project delivery to cope with current market conditions and meet stakeholder demands. The process of accelerating the completion of construction projects is referred to in the literature as the time-cost trade-off where a delicate balance between the overall cost of a project and its duration is established to achieve the desired overall project objectives (Moselhi and Esfahan, 2013). The schedule duration reduction results in a higher direct cost and a lower indirect cost. The direct costs are the costs of all resources directly used in the project exaction phase; likewise, indirect costs are incurred whether or not the productive work is accomplished. Schedule reduction results in hiring additional resources to reduce the individual activities' actual durations, which give rise to a progressive

increase of the direct project cost and a steady reduction in the indirect project cost. Accordingly, the minimization of such increased direct cost and finding the point of least-cost duration has always been of interest to researchers and professionals alike.

Accounting for resources while scheduling the compressed construction project, known as resource allocation and leveling, is another difficulty that contractors face. Research and industry used different techniques to perform scheduling that provides the shortest duration with a minimum cost. The critical path method (CPM) is one of the oldest ways that has been widely used for project scheduling in construction projects. It was developed in the late 1950s by Morgan Walker and James Kelley to avoid the costs of plant shutdowns and restarts caused by inefficient scheduling. Although CPM accounts for the time and determines critical activities to minimize project duration, it doesn't consider resource planning. Karaa and Nasr (1986) pointed out that tremendous costs can occur due to the unsmoothed utilization of a specific resource known as resource-leveling. Adjusting the number of resources in response to the time-varying requirement results in additional costs related to extra charges such as hiring and firing resources known as releasing and acquiring costs. Besides, CPM does not directly allow for activity splitting, which necessitates the stoppage of work execution in that activity due to insufficient resources. Activity splitting allows for stopping an activity and resuming it at a later date. This is very significant to enhance the project's optimal makespan when there is resource scarcity or temporary resource unavailability (Buddhakulsomsiri and Kim, 2006). Therefore, the decision is to balance the extra costs of acquiring and releasing resources incurred due to the fluctuation in their utilization over time and the activity splitting associated expenses.

The uncertain nature of the activity's time and cost adds a dimension of complexity to the already hard to solve combinatorial problem. The problem lies in assuming that a crisp (precise)

duration and cost of the activities can be pre-determined by the project managers and planners. However, construction projects are unique and seldomly entirely identical and can be affected by external and internal events such as inflation, economic and social stresses, and climate changes. This may be associated with certain activities' uncertainties, which requires scheduling methods that account for such conditions. Accordingly, researchers have successfully applied the probability theory to address the uncertain nature of construction projects. However, it's hard to record historical construction data in the absence of a mature tool for recording and retrieving the time/cost data (Zheng and Ng, 2005). Hence, it's might not applicable to determine probability distributions for costs and durations of activities. In light of this case, the Fuzzy Set (FFs) theory can effectively examine the consequences of the identified uncertainties (Zheng and Ng, 2005). Unlike a crisp set where an element takes a membership value of either 0 or 1, the value of an element's fuzzy memberships can have a value that varies between 0 and 1. Accordingly, this variation allows for more accurate representation when the set boundaries cannot be determined as crisp values.

1.2 Motivation

After two years of experience in construction and reviewing relevant literature on project acceleration, specific issues attracted my attention and appeared to have been left unanswered. Firstly, although contractors tend to consider schedule compression frequently, there is still no fully developed framework or commercially available software that can be used to perform such essential management functions. Accordingly, contractors lean on their judgment and intuition. Besides, even though there are reported methods in the literature to solve the schedule compression problem, project managers have minimal use. This phenomenon happens because these methods limit the schedule compression problem to some sort of time-cost trade-off analysis based on the

additional direct costs for crashing the project activities, without explicitly accounting for resource allocation and leveling. Besides, even though the construction activities have an uncertain nature, many researchers tended to follow the traditional way of assuming deterministic time-cost trade-off analysis (Harris, 1978). Based on the defined gaps, the problems mentioned earlier associated with real-life projects were accounted for and addressed in the proposed method.

1.3 Scope and Objectives

The main objective is to model optimized resource-constrained schedule compression; considering activity splitting, resource leveling, and uncertainty associated with project activities duration and cost. Accordingly, a set of sub-objectives are well-defined to achieve the stated primary objective as follows:

1- model uncertainties associated with activities' duration and cost;
2- incorporate more than one activity execution mode in the activity crashing with activity splitting;
3- account for resource-leveling and allocation; and
4- develop a stand-alone automated computerized tool.

1.4 Organization of the Thesis

Chapter 2 reviews the current literature in the schedule compression field while addressing the limitations and overcoming them. Chapter 3 presents the proposed method and its modules in detail and their mathematical formulations and problem constraints. The method implementation and validation are described in Chapter 4, using a recognized numerical example drawn from the literature. Chapter 5 shows the comparative analysis results between the proposed optimization

methods that involve performance evaluation and statistical comparisons. Finally, chapter 6 presents a summary of the study and concluding remarks.

2 CHAPTER TWO: LITERATURE REVIEW

2.1 General

The existing literature has offered various methods, and algorithms for the schedule optimization problem with defined objective functions aligned with the decision maker's objectives. The proposed algorithms can be classified as exact methods (linear/integer or dynamic programming), heuristic algorithms, and meta-heuristic or evolutionary algorithms. While the objective functions can be categorized as time-based, cost-based, and quality-based objective functions.

Exact methods relate the problem constraints with the objective function with some mathematical relationships and solve it with a linear or dynamic optimization technique (Davis and Heidorn, 1971). Such methods have been found convenient in modeling the time-cost optimization problem (TCO) and can offer exact solutions. However, their effectiveness descents rapidly as the project size and the resource alternatives increase as the solution becomes computationally intensive or infeasible (Dayanand and Padman, 2001). Therefore, heuristic and meta-heuristic algorithms are needed in practice to generate near-optimal schedules (Ghoddousi, Eshtehardian, Jooybanpour and Javanmardi, 2013).

Heuristic methods apply some rules for selection and exclusion to a list of generated potential solutions, leading to an optimum or near optimum solution. A typical establishment of heuristic methods is the branch and bound analysis. This method utilizes a systematic enumeration of all candidate solutions where large subsets of fruitless candidates are discarded using upper and lower estimated bounds of the optimized quantity (Christofides, Alvarez-Valdes and Tamarit, 1987). However, this method's efficiency depends heavily on the effectiveness of the used

branching and bounding algorithms. Besides, there is no universal bounding algorithm that works for all problems.

Metaheuristics and evolutionary algorithms have received the focus of most of the recent studies since the TCO problem attains a significant size as the number of project activities, resource types, and activity execution modes increases. Accordingly, the metaheuristics algorithms find, generate, or select a heuristic that may provide a sufficiently good solution to an optimization problem, especially with incomplete or imperfect information or limited computation capacity. Examples of the metaheuristics methods are the Genetic Algorithm (GA), Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO). The genetic algorithm is inspired by the natural evolution theory, which uses a heuristic search. It was first proposed by John Holland (Goldberg, 1989). In the genetic algorithm, a population is created from a candidate set of solutions. Each individual has a gene, and each gene is connected to other genes to form a chromosome, which represents a possible solution. For the generation of a new population, the algorithm uses three operators: selection operator, crossover operator, and mutation operator. In the selection phase, individuals are selected based on their fitness score towards the objective function, and their genes are passed to the next generation. The fitness score of an individual will determine the selection probability for reproduction. Accordingly, the crossover operator swaps some parent chromosomes genes, forming the offspring (new populations). In contrast, the mutation operator flips some parent chromosomes' digits to introduce useful genetic materials while producing the offspring (Zheng, Ng and Kumaraswamy, 2004)

On the other hand, Ant colony optimization is a population-based meta-heuristic method proposed by Dorigo (1992), inspired by the ant colonies' foreign behavior. The ACO aims to combine prior information about promising solutions with posterior information about the better-

found solutions to search for solutions to the combinatorial problems (Dorigo, 1992). Contrary to the GA, ACO resembles the particle swarm optimization (PSO) in that they utilize previous information of the entire colony rather than the previous generation only (Li and Zhang, 2013). However, PSO is inspired by the swarm's choreography with the same natures, unlike ACO, based on a population of individuals with different structures but a common goal. Finally, ACO, unlike GA and PSO, utilizes some extra heuristic information in the search course, which may help and speed up finding the optimality. (Li and Zhang, 2013)

Previous methods that attempted to optimize the resource-constrained project scheduling or perform schedule compression in the literature can be classified into four categories: (1) Time-Cost optimization models; (2) Deterministic resource-constrained scheduling models; (3) Resource-constrained scheduling under uncertainty models; and (4) Resource allocation scheduling models.

The objective of time-cost optimization models (TCO) is to compress the project schedule while maintaining a project budget (Zheng, Ng and Kumaraswamy, 2005). The objective of deterministic resource-constrained scheduling models is to develop optimal schedules subject to resource availability and project duration minimization or confinement within given deadlines (Kaiafa and Chassiakos, 2015). In contrast, the resource-constrained scheduling objective under uncertainty models analyzes the project schedule in an uncertain environment. The resource allocation scheduling problem belongs to the class of NP-hard problem and is an extension of the Multi-mode resource-constrained project scheduling problem (MMRCPSP). The resource allocations models' objective is to minimize the project duration and cost and account for resource-leveling and availability. Resource leveling aims to reduce the peaks and valleys in the resource

usage profile while considering the project deadline to achieve the most efficient resource consumption and project duration.

This chapter reviews the proposed time-cost optimization models, deterministic resource-constrained models, resource allocation scheduling models, and resource-constrained scheduling under uncertainty models. Moreover, the main limitations of the proposed models are described in the summary section.

2.2 Time-Cost Optimization Problem

Moselhi and Esfahan (2013) developed a novel method to deal with the trade-off between time and cost. A modified multiple binary decision method (MBDM) and a multi-attributed schedule crashing algorithm were proposed to reduce the project schedule. They utilized the MMBD as a multi-criteria decision support method to set priorities for the activities. Accordingly, an iterative crashing method was used to progressively reduce project duration in search for a least-cost duration based on the assigned priorities. Moselhi and Alshibani (2013) used iterative crashing subject to the fuzzy costs of activities and contractor's judgment. The model used set priorities and weights from the experts to optimize schedule compression and produce the best compression plan based on the least cost-duration and contractor's judgment. Feng, Liu and Burns (1997) demonstrated the use of the convex hull method in solving the multi-objective time-cost trade-off based on the principles of the genetic algorithms. They developed a computer program that can efficiently execute the proposed methodology by finding non-dominated solutions. Zheng, Ng and Kumaraswamy (2004) modified the traditional adaptive weight method. The proposed algorithm establishes a combination of time-cost solutions by utilizing previous iteration information to adjust the next search's scope based on the current population's performance in obtaining global optimization. Afshar, Ziaraty, Kaveh and Sharifi (2009) utilized a multi-colony non-dominated

archiving method ACO where each of the ant colonies is assigned to each of the time-cost trade-off problem (TCTP) targets. Aminbakhsh and Sonmez (2016) developed a particle swarm optimization for medium and large-scaled TCTPs. Their efficient developed methodology proved to outperform the previously proposed methods in solution quality and computation time, especially for large-scale projects. Toğan and Eirgash (2018) used a Teaching- Learning Based Optimization incorporated with the Modified Adaptive Weight method to find a set of Pareto front time-cost trade-off solutions. According to the authors, the proposed algorithm works effectively to find optimal or near-optimal solutions for TCTP in the construction engineering and management field than the previously proposed metaheuristic methods.

2.3 Deterministic Resource-Constrained Project Scheduling Problem

The assumption in the primary form of the resource-constrained project scheduling problem is that each activity can be performed by one method within a determined processing time with one renewable resource. The extended way of the RCPSP is the multi-mode resource-constrained project scheduling problem (MMRCPSP) in which several techniques (alternatives) are available to execute the project activities. Each execution mode has a specific processing duration and resource requirement (Hartmann and Briskorn, 2010).

MMRCPSP was first introduced by Elmaghraby (1977) where the author assumed that an activity j must be performed by a mode $m \in M_j$ and must be processed in the same way until completion. Brucker *et al.* (1999) proposed the branch and bound method to solve the MMRCPSP. The searching tree reduction schemes enhanced the branch and bound methodology's basic enumeration scheme by Sprecher and Drexl (1998). Similarly, Heilmann (2003) used a branching strategy where the branching rule was selected dynamically. He successfully managed to determine a solution that represents the modes and start times simultaneously. On the other hand,

Tao and Tam (2013) employed the Levenberg Marquardt plus Universal Global Optimization method to optimize MMRCPSP with time, cost, and quality objectives.

Along with the different exact solving methods, some researchers used heuristic algorithms to solve the problem in a less timely manner since heuristic methods allow selection between competing for limited resources. Heilmann (2001) proposed an integrated approach based on a multi-pass priority with back planning embedded in random sampling. Erenguc, Ahn and Conway (2001) followed a heuristic method to minimize the cost of a generated initial feasible schedule using the proposed six rules. The six rules are based on the selected execution mode, including a different combination of labors and equipment. The rules included precedence relationships, resource capacities for each project period, and assigning only one mode to each activity. Singh (2014) utilized priority rules and the AHP method based on an integrated method to minimize project duration and delay penalty simultaneously.

To cope with the complex nature of the construction project activities and to have more realistic scenarios representing the construction industry, researchers explored using metaheuristic methods to analyze more activities with multiple execution modes. Chen and Weng (2009) developed a two-phase genetic algorithm that resolves the time-cost trade-off and resource-constrained scheduling problem. To jointly optimize both objectives, a GA-based time-cost tradeoff analysis is used to select modes to balance the cost and time, followed by utilizing a GA-based resource scheduling model to consider the project constraints while generating the feasible schedule. Coelho and Vanhoucke (2011) split the MRCSP into two steps to improve solution quality and speed. They presented an algorithm that has a mode assignment step and a mode project scheduling step. The first step is solved by a fast solver, whereas the second step is solved using a GA. Baradaran, Ghomi, Ranjbar and Hashemin (2012) analyzed a hybrid genetic algorithm to

solve the MRCSP problem in PERT networks. Their analysis was based on the scatter search and a path relinking algorithm. Menesi and Hegazy (2015) utilized constraint programming (CP) as an advanced mathematical optimization technique that suits scheduling problems. The IBM ILOG modeling software and its CPLEX-CP solver engine have been used to develop a CP optimization model for the multi-mode schedule compression problem. Although the proposed model is fast and provides a near-optimum solution for projects with hundreds of activities within minutes, it optimized only the project duration with resources constraint.

Resources categories were first defined by Slowinski (1980) as renewable and non-renewable resources based on their availability. Renewable resources represent labor, machinery, and equipment and non-renewable resources represented money. Accordingly, a resource is renewable if the amount of that resource is constant at every moment of the project duration, and a non-renewable resource if only its total consumption (integral availability up to the project deadline or a given moment) is limited (Węglarz, Józefowska, Mika and Waligóra, 2011). Li and Zhang (2013) considered the renewable and nonrenewable resources in an ant colony optimization algorithm to solve the resource-constrained scheduling problem. Azizoglu, Çetinkaya and Pamir (2015) developed a linear programming relaxation-based heuristic solutions algorithm implemented in a project scheduling problem with a single non-renewable resource. Altintas and Azizoglu (2020) published one of the latest papers on the multi-mode resource-constrained and discrete time-cost trade-off problem. They optimized the activity execution modes with a single non-renewable resource released at specified times and specified quantities in a progressive way. Renewable and non-renewable resources were studied by Chaleshtarti and Shadrokh (2014) as an extended form of the resource-constrained scheduling problem. They proposed a branch and cut algorithm with some techniques that shorten the models' size related to the nodes and some

fathoming rules that lessen the number of nodes. The developed method specifies the lower bounds for the problem in any middle stage of the solving process that is useful to deal with large instances, where solving processes take a long time. A bi-objective multi-mode resource-constrained scheduling problem is studied with renewable and non-renewable resources that focus on maximizing the net present value while minimizing the project duration by considering the discounted cash flow (Tirkolaee, Goli, Hematian, Sangaiah and Han, 2019). The proposed model was constructed using the non-dominated sorting genetic algorithm and multi-objective simulated annealing algorithm and implemented by CPLEX solver of GAMS software. The Ɛ-constraint method, a well-known approach to deal with the multi-objective problems which can generate Pareto solutions, was constructed to validate the developed metaheuristic methods. Single renewable resources were considered in a resource-constrained genetic algorithm by Kadam and Kadam (2014). The authors claim that their model is more effective for the RCPSP than the existing optimizations algorithm. Multiple projects scheduling problem was investigated by Afruzi, Aghaie and Najafi (2018). This problem involves multiple projects with different importance weights, activities with uncertain durations, predefined assigned due dates, and renewable resources with a sharing policy. The model's main objective is to find an optimal structure containing all the projects such that the maximum weighted differences between the project's finish times and their assigned due dates will be minimal while transferring the resources between the activities based on the resource sharing policy.

2.4 Resource-Constrained Scheduling under uncertainty Problem

Project scheduling success depends heavily on having an accurate baseline schedule that allows better control and easy planning. However, to have a precise baseline schedule, the scheduling timetable must be obtained by contractors before starting the project with accurate

information. The availability level of the required information can be full or limited. In the case of having limited information, the project schedule must be analyzed in an uncertain environment. The resource-constrained scheduling problem is faced with significant uncertainties such as activities that might take a shorter or longer duration than the estimated amount, resources that might not be available at the time of need, and unpredictable natural disasters, weather, and climate change. Some contractors use historical data to account for uncertainties; however, historical data is always limited in construction projects. The historical data limitation is due to the lack of a mature tool for recording and retrieving the data needed to represent the uncertainty as a probability distribution for the duration and cost (Zheng and Ng, 2005). Accordingly, the literature revealed that the fuzzy theory used to model uncertainty in the construction field are more appealing and preferred than random numbers (Herroelen and Leus, 2005). Fuzzy set theory (FSs) was introduced by Zadeh (1965).

Eshtehardian, Afshar and Abbasnia (2009) considered the uncertain nature of the time-cost optimization problem by embedding the fuzzy structure of the uncertainties in the total direct cost. The project manager's acceptable risk level is defined through a defuzzification method, α-cut method, for which a separate Pareto front with a set of non-dominated solutions has been developed. The α-cut concept shows the level to which project experts want to encompass uncertainties in calculations and is directly associated with their courage and confidence in their decision-making. Associated fuzzy costs for different values of α-cut are ranked by comparing the alternative set of options for any assumed project duration. Finally, an appropriate GA is used to develop a multi-objective fuzzy time cost optimization model. Two concepts of time-cost trade-off, resource leveling, and allocation have been embedded in a stochastic multi-objective optimization model by Zahraie and Tavakolan (2009). The authors attempted to minimize the total

project time, cost, and resource moments simultaneously while considering the uncertainties. In the proposed research, the fuzzy set theory was applied to increase decision-makers' flexibility while using the model outputs. The Left and right dominance ranking method was used in an ant colony optimization algorithm by Kalhor, Khanzadi, Eshtehardian and Afshar (2011) to find the non-dominated solutions for the stochastic time–cost trade-off optimization problem. The proposed ranking method defines the decision maker's optimism using the β concept to rank the fuzzy numbers based on the left and right dominance. Simultaneously, the model searches for non-dominated solutions considering the project's total duration and total cost as two objectives. Birjandi and Mousavi (2019) used a hybrid heuristic and metaheuristic stochastic algorithm to consider the uncertainty in the resource-constrained scheduling problem. A heuristic algorithm based on distribution rules was used to generate high-quality initial solutions to solve the trade-off between cost and time based on multiple routes. The heuristic distribution rules were used instead of random solutions to generate appropriate and initial solutions to speed up reaching the near-optimal solutions. Then particle swarm optimization (PSO) algorithm was used to change and assign an appropriate route from available routes for flexible activities. Lastly, a genetic algorithm is proposed to generate the best solutions from the PSO's generated routes. Salama and Moselhi (2019) utilized the center of area method as a defuzzification method to solve the stochastic multi-objective optimization for repetitive scheduling with activity interruptions. Their model design integrated six modules: schedule calculations using the integration of linear scheduling method and critical chain project management, uncertainty and defuzzification module using fuzzy set theory, cost calculations module that considers direct and indirect costs, delay penalty and work interruptions cost, module for identifying multiple critical sequences and schedule buffers and reporting module and finally a multi-objective optimization module using genetic algorithm.

2.5 Resource Allocation Scheduling Problem

The resource allocation scheduling problem is one of the most challenging projects management problems. It is a combinatorial problem with multiple objectives (project duration confinement, resource-constrained allocation, resource leveling, and activity splitting). The problem's complexity and size grow exponentially as the resource types, activities, and execution modes increase. Resource leveling aims to provide a smooth resource utilization that prevents day-by-day resource fluctuations (Kaiafa and Chassiakos, 2015).

One of the earliest attempts to solve the resource leveling was by Harris (1990), where the author proposed a minimum moment method for resource-leveling. Hiyassat (2001) modified Harris's original method to apply it to multiple resource projects (multiple resource leveling). Tawalare and Lalwani (2012) improved the selection criterion of activities that will be moved or reloaded within the project duration and the resource histogram. On the other hand, Christodoulou, Ellinas and Michaelidou-Kamenou (2010) modified Harris' method as an entropy maximization problem where activity stretching is allowed, and resource allocation solutions are improved to previous methods. Abdel-Basset, Ali and Atef (2019) proposed a model to minimize the costs of daily resource fluctuations using the precedence relationships during the project completion while overcoming the ambiguity caused by real-world problems by designing a resource leveling model based on the neutrosophic heuristic procedure. The Multiple attribute decision-making model was utilized to find non-dominated solutions generated from a traditional genetic algorithm to solve the resource allocation scheduling problem (leu and Yang, 1999). The authors developed a multi-criteria optimization model for construction schedule based on a genetic algorithm, which incorporated the time-cost trade-off, resource-constrained allocation, and the limited resource-leveling models. The model was implemented in two phases. In phase 1, non-dominated solutions

were found for time and cost functions under resource constraints. In phase 2, resource leveling was performed, and the starting dates of activities were determined. Ghoddousi, Eshtehardian, Jooybanpour and Javanmardi (2013) improved Leu's model by solving the multi-mode resource-constrained project scheduling, discrete time-cost trade-off and, resource leveling problems simultaneously. NSGA-II was used to find a non-dominated solution for the optimal time, cost, and resource moments. (Hegazy and Menesi (2012) developed an optimization model that uses cycles of crashing for lowest-cost critical activities (i.e., stepwise time/cost trade-off process) and resolves any resource over allocation within each time/cost trade-off cycle. Hegazy (1999) minimized the total project duration and appropriate moments simultaneously under resource constraints using a near-optimal solution generated by a genetic algorithm. Hegazy and Ersahin (2001) developed Schedule optimization concerning time, cost, and resource constraints using a spreadsheet model. Resource leveling was extended by Roca, Pugnaghi and Libert (2008) to account for the minimization of the project span and the leveling of resource usage over time and produce non-dominated solutions.

The resource allocation and time buffering problems were recently introduced as a bi-objective robust project scheduling model (Liang, Cui, Hu and Demeulemeester, 2019). The authors managed to fill the previous studies' gaps where integration between resource allocation and time buffering was proposed. Besides, they considered the project due date and the activity start times against disruptions during execution. Xu, Hao and Zheng (2020) developed a logistic task resource allocation model for the multi-stage resource leveling problem in sharing logistic networks known as multi-objective resource-constrained project scheduling problems. The proposed model considers the total cost and duration for sharing logistics network and improves the intra-stage efficiency and inter-stage stability for resource providers. The authors provided a

multi-objective artificial bee colony algorithm with adaptive neighborhood rules that improve the slow convergence, weak local search, and easy-to-precocious of the traditional artificial bee colony algorithm, NSGA-II, and multi-objective particle swarm optimization.

The basic assumption in the traditional resource-constrained project scheduling problem (RCPSP) is that activities cannot be interrupted once started. Accordingly, the number of resources needed by activity A will be held during that activity and cannot be used for other activities until activity A finishes. However, in real case projects, some activities might be interrupted due to equipment repairs or insufficient resources at any period of the project. The activity splitting was raised for the first time in 1988 by Kaplan; He stated that construction activities could be stopped and resumed without additional costs. Peteghem and Vanhoucke (2010) solved the Preemption multi-mode resource-constrained project scheduling problem by applying a bi-population genetic algorithm that uses two populations and extends the serial schedule generation scheme by introducing a mode improvement procedure. Preemption without a penalty was discussed by Moukrim, Quilliot and Toussaint (2015). An effective branch-and-price algorithm was used to minimize the project duration based on minimal interval order enumeration involving column generation and constraint propagation. On the other hand, preemption with a penalty and the earliness-tardiness cost were introduced by Afshar-Nadjafi (2014). The author developed a mixed integer programming model to minimize the total project cost, considering earliness-tardiness and preemption penalties. The model assumed an activity could be restarted after being interrupted in a discrete point of time with a constant setup penalty and without setup time. On the contrary, Li, Lai and Shou (2011) proposed a hybrid particle swarm optimization model that permits activities to be interrupted only once during the whole project. The model consisted of two schedule generation schemes that decode the designed four types of particle representations. Cheng, Fowler,

Kempf and Mason (2015) introduced the non-preemptive activity splitting to deal with renewable resources' varying capacity. In their method, an activity can be interrupted one or more times after starting whenever the resource levels are insufficient and will resume in the next eligible processing period. Hariga and El-Sayegh (2011) developed an optimization model for resource-leveling that allows activity splitting and minimizes its associated costs. Their objective was to level resources to provide a trade-off between the extra cost of acquiring and releasing resources versus the additional cost of activity splitting. Hariga, Shamayleh and El-Wehedi (2016) proposed a mixed-integer linear programming model to provide a least-cost project schedule with well-adjusted resource utilization and reduced project duration. The proposed method integrated the activity crashing and leveling problems with the possibility of splitting activities.

2.6 Summary

After a thorough literature review on schedule compression and project scheduling, some gaps and possible improvements were identified for a more realistic and efficient project scheduling. The capabilities and limitations of the most recent relevant references to this study are summarized in Table 1. The joint research gaps that were encountered while reviewing the literature can be summarized in the following points:

1. Unconstrained-Resource scheduling methods.
2. Deterministic resource-constrained scheduling methods.
3. Scheduling methods that don’t account for activity splitting.
4. Scheduling methods that don’t account for the extra costs of acquiring and releasing resources such as hiring and firing resources represent the resource utilization profile variation.

5. Scheduling methods that don't account for delay penalty/ opportunity cost.

Table 1: Capabilities and limitations of the most recent and relevant references

Criteria	**Reference**							
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Optimization methods	NSGA-II	ACO	BIP	GA	PSO	MILP	MAWA-TLBO	MILP
Multi-objective optimization	✔	✔	✔	✔	✔	✔	✔	
Consider Cost	✔	✔		✔	✔	✔	✔	
Deterministic			✔		✔	✔	✔	✔
Considered the activities' uncertainty	✔	✔		✔				
Resource-constrained	✔							✔
Resource leveling	✔		✔			✔		
Consider delay penalty/ bonus				✔				
Allow activity splitting			✔	✔		✔		
Consider resource acquiring and releasing costs			✔			✔		

NOTES: (NSGA-II) Elitist Non-Dominated Sorting Genetic Algorithm, (ACO) Ant Colony Optimization, (BIP) Binary Integer Programming, (IOE) Branch-and-Price Algorithm, (PSO) Particle Swarm Optimization, (EA) Evolutionary-Based Algorithm, (MAWA-TLBO) Modified Adaptive Weight Approach and Teaching Learning Based Optimization, (MILP) Mixed Integer Linear Programming; (1) Zahraie and Tavakolan, 2009, (2) Kalhor, Khanzadi, Eshtehardian and Afshar, 2011, (3) Hariga and El-Sayegh, 2011, (4) (Moselhi and Salama, 2019), (5) Aminbakhsh and Sonmez, 2016, (6) Hariga, Shamayleh and El-Wehedi 2016, (7) Toğan and Eirgash 2018, (8) Altintas and Azizoglu, 2020.

To address the above needs, this study proposes a resource-constrained schedule compression method to account for resource planning and project scheduling under uncertainty simultaneously while accounting for activity splitting. The method's objective is to minimize the project duration and cost, including direct cost, indirect cost, and delay penalty, and strike a balance between the cost of acquiring and releasing resources on the one hand and the cost of

activity splitting on the other hand. The method's objectives are subjective to the duration, network logic, resource balance and activity splitting constraints. To the best of our knowledge, the proposed method is the first to address the resource-constrained schedule compression under uncertainty and resource-leveling problems simultaneously with the possibility of splitting activities.

3 CHAPTER THREE: DEVELOPED METHOD

3.1 General

This study provides an integrated method for the resource-constrained project scheduling problem in an uncertain environment with the possibility of selecting an optimal solution from a developed set of non-dominated solutions. The proposed method utilizes the fuzzy set theory (FSs) for modeling uncertainty and the genetic algorithm (GA) for optimization. By combining GA with FFS, different sets of Pareto solutions are identified by employing different risk acceptance levels. The user's affordable risk level is transformed into crisp values through the α-cut concept, for which a separate Pareto front has been developed. The integration of the fuzzy concept within the genetic algorithm optimization procedure facilitates the decision-support process by selecting a specified risk level and employing the associated Pareto front. The developed method expands upon the work discussed in the literature review chapter by providing an integrated resource-constrained project scheduling method to optimize project duration and cost through the resources and cost of the execution modes assigned to project activities. The integrated method accounts for project cost and resource-leveling based on costs and resources imbedded in these modes of execution. The developed method has five main modules that support two different optimization methods, as shown in Figure 1.

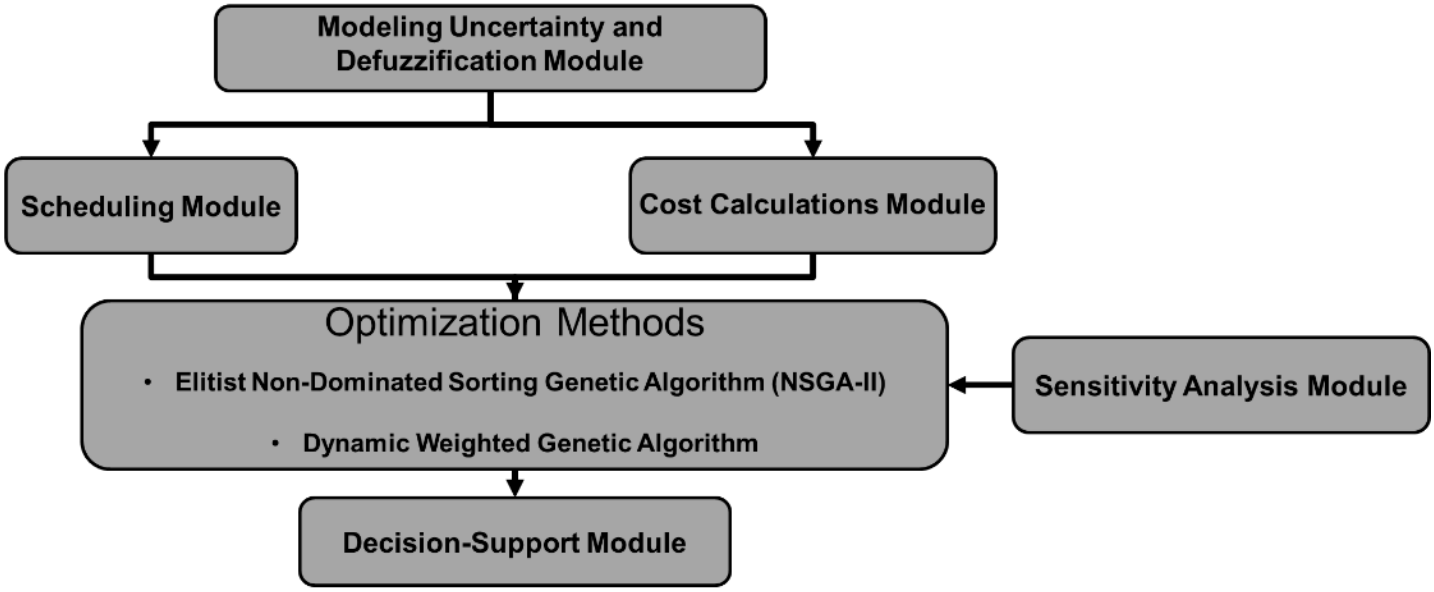


Figure 1 Developed modules

The method steps to generate an optimal schedule and project cost using the developed modules and optimization methods are summarized in the flow chart, shown in Figure 2.

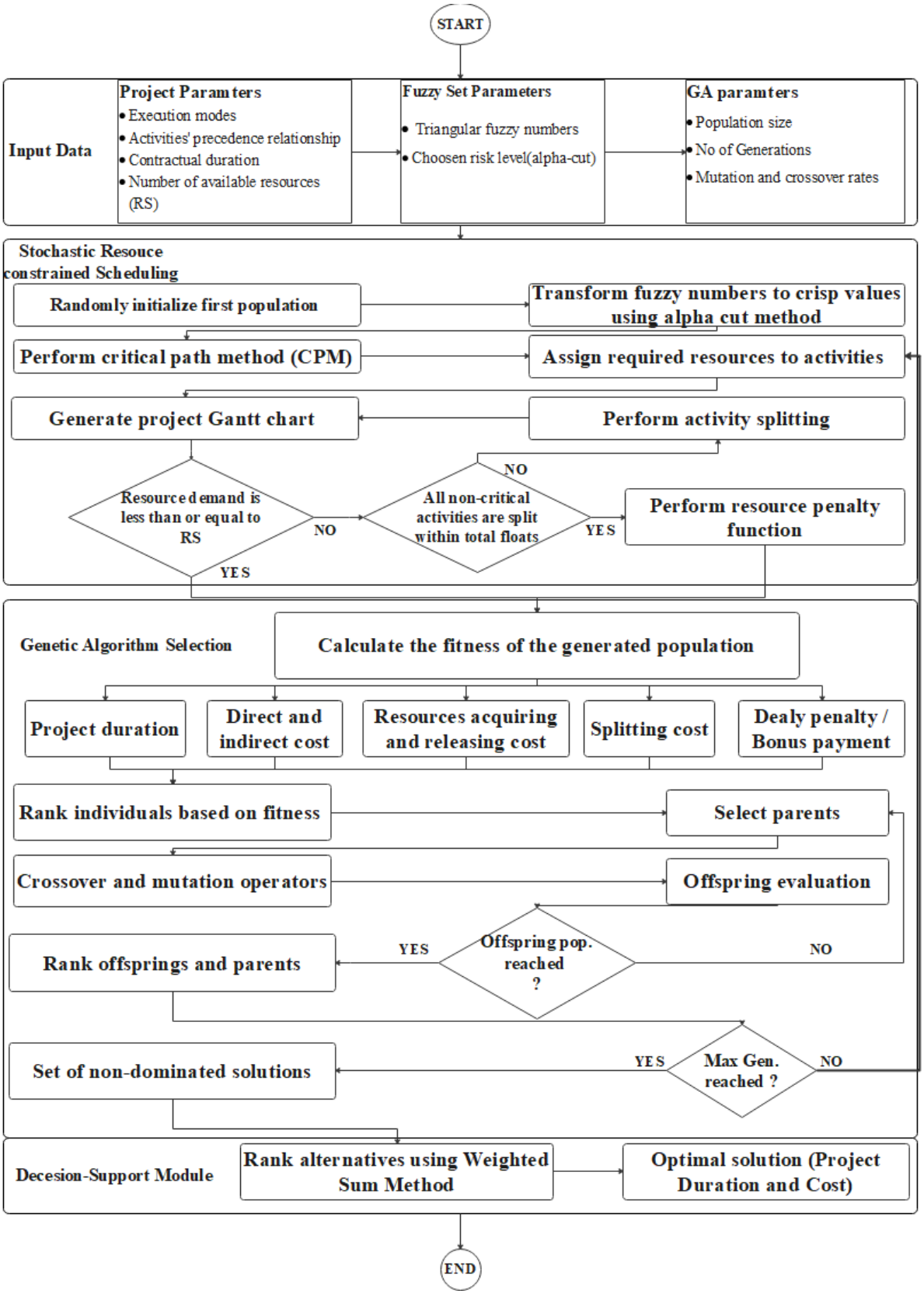
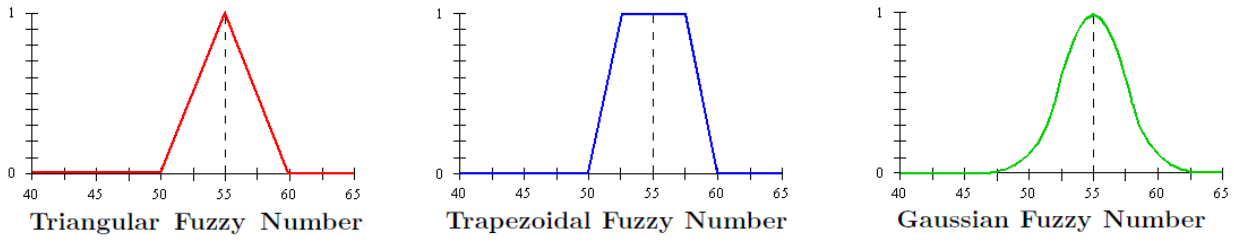


Figure 2 Flow chart of the proposed method

3.1.1 Modeling Uncertainty and Defuzzification Module

The first module aims to model the uncertainty of the construction activities' duration and cost using the fuzzy set (FFs) theory and transform fuzzy input variables into crisp values. The FFs comprise two steps; fuzzification and defuzzification, where fuzzification is changing a real scale value into a fuzzy value, and defuzzification is replacing the fuzzy variable for a crisp one. Fuzzy set qualifies as fuzzy numbers if they are normal, convex, and bounded, where a fuzzy number can be represented by triangular, trapezoidal, or gaussian shapes, as seen in Figure 3. Since Klir, Wang and Harmanec (1997) pointed out that the fuzzy membership function's used shape does not significantly show sensitivity to most FSs applications, triangular fuzzy numbers (TFN) will be used for simplicity to represent the uncertainty of the project activities. On the contrary, Defuzzification methods have been proposed in the literature and the most popular technique is the center of gravity, which involves some sort of weighted averaging over its membership function. However, a fuzzy membership can also be represented by its α -cut where each α -cut indicates a confidence level at its corresponding certitude level α, as shown in Figure 4. The alpha-cut method provides a decision analysis tool for the user to model acceptable risk levels.

Figure 3 Types of Fuzzy number representations (Ouma, Yabaan, Kirichu and Tateishi, 2014)



By adopting FSs theory, the developed module allows the user to input the activities' duration and cost with a fuzzy triangular representation. TFN is represented by three points (a_1,

a_2, a_3) on the universe of discourse, representing the minimum, most likely, and maximum values, respectively, as seen in Figure 4. Simultaneously, an alpha-cut method is used as a defuzzification method to find the project duration and cost-based on the user preference alpha-cut value that determines the acceptable risk level. The used alpha-cut method takes the average of the left and right endpoints (a_1 and a_3), known as minima and maxima of the alpha-cuts, respectively, to find the crisp values. A sample code on the modeling uncertainty and defuzzification module is provided in Appendix I. The activities' triangular fuzzy representation provided by the user is used to calculate the project duration based on the alpha cut method. The developed code for the modeling uncertainty and defuzzification module utilized some functions from a python library called Skfuzzy.

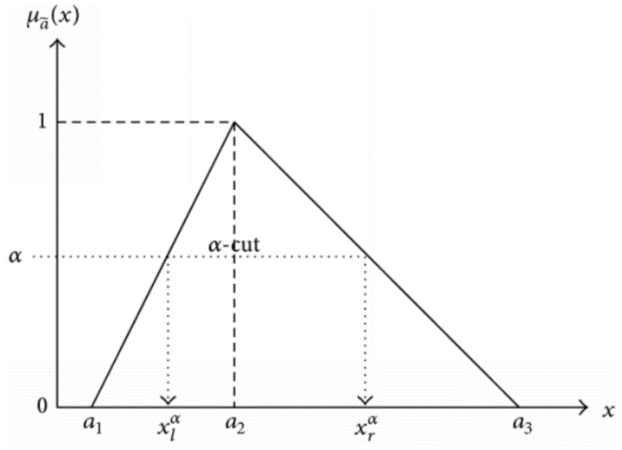


Figure 4 Triangular fuzzy number representation (Martin & Klir, 2007)

3.1.2 Scheduling Module:

For the scheduling module, the user is asked to identify the normal and crashed durations, related costs and resource requirements, precedence relationships, and lags/Lead for each activity, as shown in the user terminal in Figure 5.

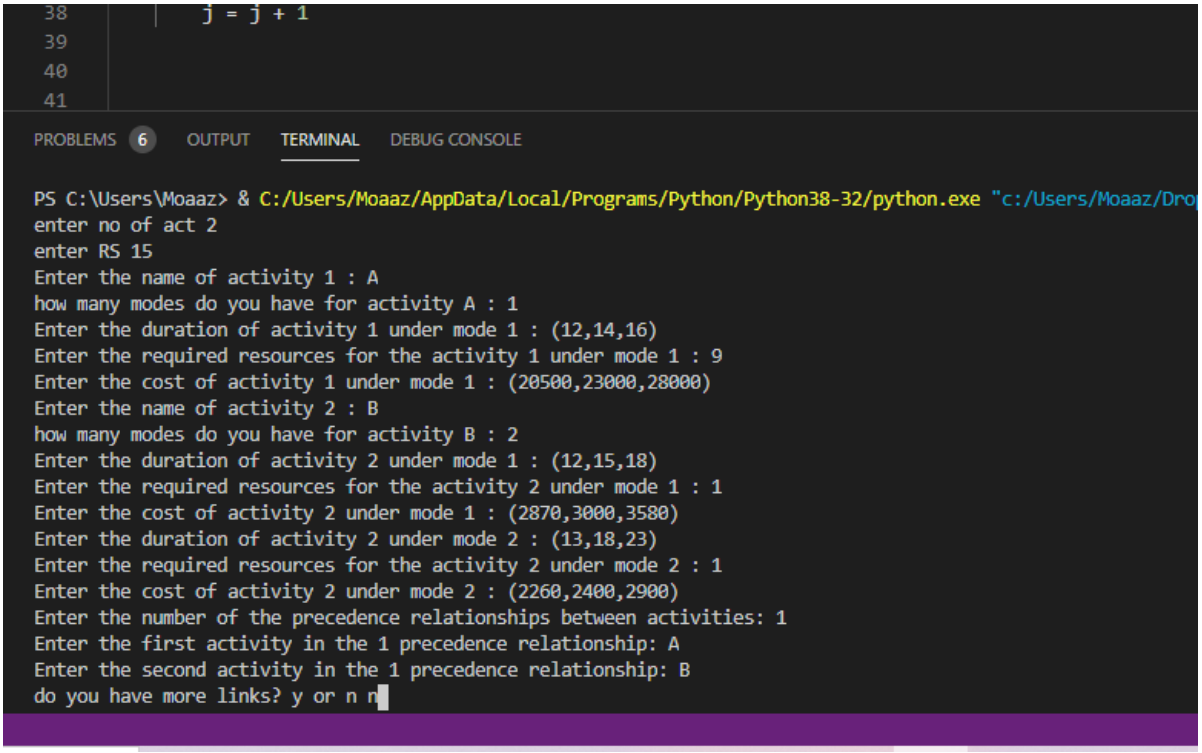


Figure 5 User terminal

The developed scheduling module utilizes the user inputs to calculate the project duration and generates a feasible project schedule based on the following steps.

1- Based on the precedence relationships between the activities, the forward and backward calculations are performed to determine the critical and non-critical activities, which result in a critical path(s) and project duration.
2- The earliest start time, latest start time, earliest finish time, latest finish time, and total float are calculated for each activity.
3- Initially, critical and non-critical activities are scheduled at their earliest start time.
4- The resource demand is calculated for each period of the project based on the activities' resource requirements.
5- Non-critical activities are rescheduled within their early start and late finish based on resource availability and resource demand.

6- An activity splitting function, which will be explained in the next paragraph with an example, is used to split non-critical activities if the resource demand is higher than the availability at any period of the project.

7- Finally, the number of acquired and released resources for each period t of the project are calculated based on the resource demand on period t and t+1. The calculated number of acquired and released resources represents the variation of the resource's utilization along the project's span.

Sample code on the scheduling module is provided in Appendix I. A Python critical path library, Criticalpath, was utilized to calculate the early, late start, early, and late finish and define the project duration's critical activities. The critical path library was then modified to assign the required resources to the project activities and accordingly generates a Gantt chart for the project that calculates the resource demand for each project period and performs non-critical activity splitting based on the availability of the resources.

Scheduling Module's Splitting Function

To demonstrate the scheduling module's splitting function, we will use an illustrative example that is an extension to the one used in (Son and Mattila, 2004) for the multiple resource case. The type of resource that is used by all activities was a single machine. The input data and project network are shown in Table 2 and Figure 6, respectively.

Table 2: Example input data

Activity	Duration	Resources	ES	LS	EF	LF	TF
A	2	2	0	2	0	2	0
B	3	4	2	5	2	5	0
C	2	2	5	7	5	7	0
D	3	1	7	10	7	10	0
E	3	4	10	13	10	13	0
F	2	6	13	15	13	15	0
G	4	4	0	4	3	7	3
H	3	5	4	7	7	10	3
K	2	2	2	4	7	9	5
L	4	2	4	8	9	13	5

As seen in Figure 6, the non-critical activities are G, H, K, and L, which can be split in case of insufficient resources. However, activities A, B, C, D, E, and F are critical activities that should not be interrupted to keep the original project duration.

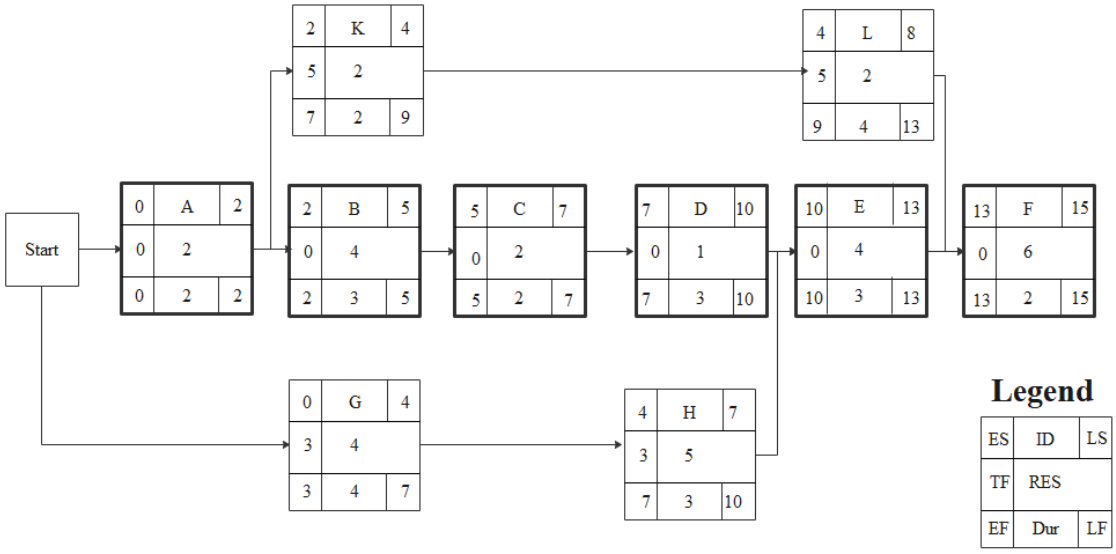


Figure 6 Example project network

The Gantt chart for the previously mentioned example before splitting is shown in Figure 7. The project schedule has nine available resources for 15 days period where the resource demand was higher than the available resources in periods 3, 4, and 5. The 1's and 0's indicate when an

activity is active during its total float. For example, activity G has a total float of 7 days with four days duration, where it was active from period 1 to 4. However, it can be split within its floats based on the resource availability on these periods without affecting the project schedule.

Activity/Periods	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	1	1													
B			1	1	1										
C						1	1								
D								1	1	1					
E											1	1	1		
F														1	1
G	1	1	1	1	0	0	0								
H				1	1	1	1	0	0	0					
K		1	1	0	0	0	0	0							
L				1	1	1	1	0	0	0	0	0	0		
Resources Demand	6	9	10	14	10	9	9	3	3	3	4	4	4	3	3
Acquired Resources	6	3	1	4	0	0	0	0	0	0	1	0	0	0	0
Released Resources	0	0	0	0	4	1	0	6	0	0	0	0	0	1	0
Available Resources	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9

Figure 7 The example initial Gantt chart

A Gantt chart and resource utilization profile for a feasible project schedule with splitting are generated and shown in Figures 8 and 9. The resource demand was equal to or less than the available resources. As illustrated in the Gantt chart, activities G and L were interrupted from periods 3 to 5 and 6 to 9, respectively, since y_{tj}, binary variable equals to one when activity j is progressing at time t; t= $ES_j \; to \; LF_j$ and zero otherwise, were equal to zero in these periods. Although the generated schedule is feasible regarding resource availability, it can be observed from the resource utilization profile that such a schedule resulted in inefficient resource utilization. Lastly, it can be noted that activity H was not split; however, its starting time was adjusted to start

on period 6 rather than period 4 to satisfy the resource availability in periods 4 and 5. In this case, splitting costs won't be imposed since the activity wasn’t started yet.

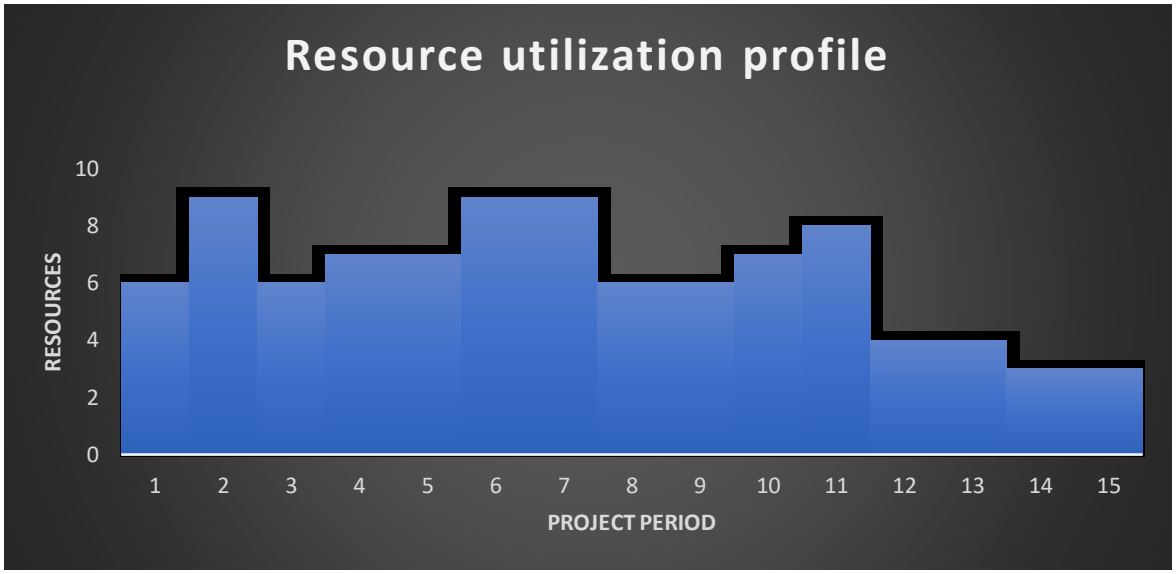


Figure 8 The example resource utilization profile

Activity/Periods	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	1	1													
B			1	1	1										
C						1	1								
D								1	1	1					
E											1	1	1		
F														1	1
G	1	1	0	0	0	1	1								
H				0	0	1	1	1	1	0					
K		1	1	0	0	0	0	0							
L				1	1	0	0	0	0	1	1	0	0		
Resources Demand	6	9	6	7	7	9	9	6	6	7	8	4	4	3	3
Acquired Resources	6	3	0	1	0	2	0	0	0	1	1	0	0	0	0
Released Resources	0	0	3	0	0	0	0	3	0	0	0	4	0	1	0
Available Resources	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9

Figure 9 The example final Gantt chart

3.1.3 Cost Calculations Module

For the cost calculations module, the user is asked to identify the contractual duration, activity splitting cost per split, indirect cost per day, resources releasing and acquiring costs per resource, and the delay penalty/bonus payment per day. The cost calculations module uses the user inputs and the scheduling module output to find the total project cost based on the following components:

1- Direct cost; the summation of the activities' costs.

2- Indirect cost; the project's indirect cost rate multiplied by the duration of the project.

3- Splitting cost; the number of splits occurred multiplied by the cost of one split, representing the extra costs associated with shutting down the activity and later restarting it.

4- Costs of acquiring and releasing resources; the number of acquired and released resources for each period of the projects multiplied by the acquiring and releasing cost rate, which represents the extra costs associated with hiring and firing resources.

5- Delay penalty/Opportunity cost; the difference between the contractual duration and the actual schedule duration multiplied by the bonus payment or the delay penalty, representing the advantage of finishing the project before or after the planned date.

3.1.4 Sensitivity Analysis Module

Since the optimization problem is solved with the genetic algorithm, the solution quality might be influenced by the GA parameters such as the number of generations, population size, mutation, and crossover rates. Usually, the user selects these parameters with no guidelines for what values will result in a better solution. Hence, the sensitivity analysis module is developed to establish the procedures for choosing the parameters' values and study the possible effects of these

parameters and their interaction on the solution's quality. Therefore, this module identifies the values of the GA parameters that produce quality solutions. The developed sensitivity module is case dependent which is very useful for optimization problems where the knowledge about the optimal GA parameters is unknown.

3.1.5 Optimization Methods

Optimization is a branch of operations research that can help in better decision-making when solving problems. Optimization aims to quantitatively minimize or maximize a goal by determining the optimum solution or set of solutions while respecting user-defined constraints. (Elmasry, Zayed and Hawari, 2019). The Genetic algorithm was chosen as it is widely used in the time cost optimization field. The genetic algorithm's population structure for the developed optimization method is shown in Figure 10.

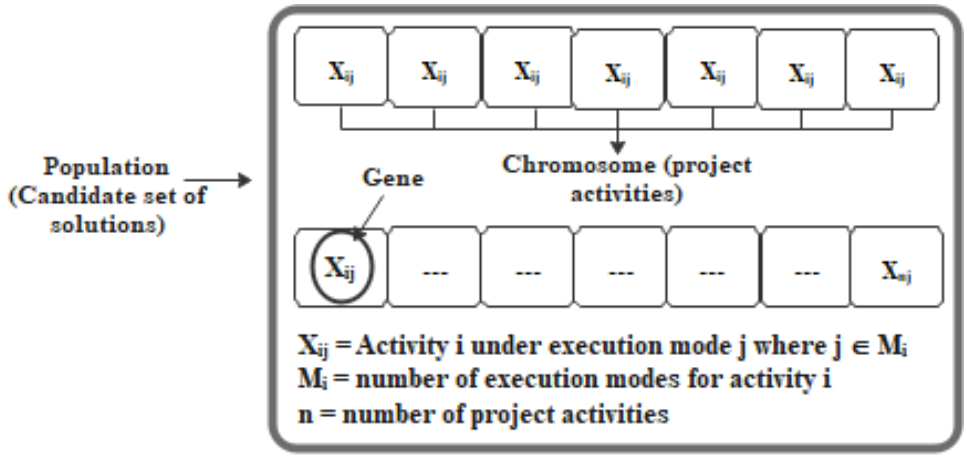
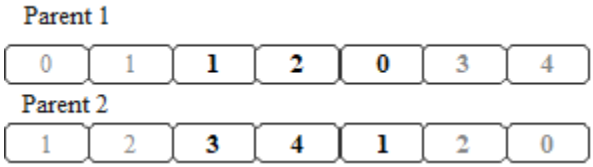


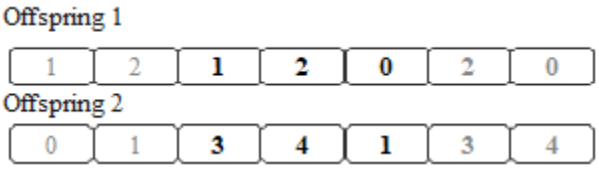
Figure 10 Description of the Genetic algorithm population

The used crossover operator in the developed optimization method is the partial mapped crossover, an extension of the two-point crossover. This crossover method was proposed by Lingle at the first conference proceedings on genetic algorithm in 1985 to solve the illegitimacy caused by the simple two-point crossover by preserving the parents' elements' order and position. The following steps explain the partial mapped crossover:

I. The crossover substring is randomly selected

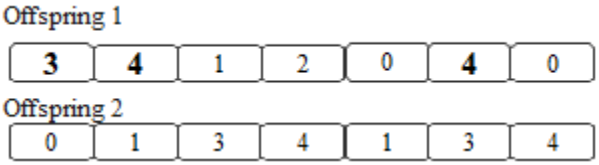


II. Substrings exchanged between parents



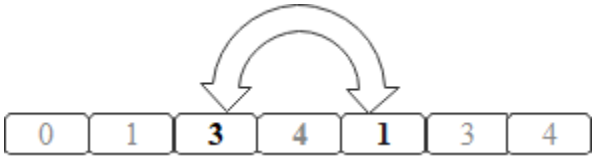
III. Substring mapping (searching for the elements in the substring that hasn't been copied yet in step 2)

IV. Replacement of doubled genes outside the substring based on the substring mapping



In the mutation operation, a simple reciprocal mutation operator is used. Two positions are randomly selected, and genes swapped on these positions. The following steps explain the reciprocal mutation operator:

I. Two positions are randomly selected



II. Genes on these positions are swapped

Resource-Constrained Optimization Problem

The resource-constrained problem was handled in the developed method by the following functions:

1- Rescheduling non-critical activities within their floats.
2- If the resource demand is still higher than the available resources, splitting non-critical activities within their floats.
3- If the resource demand is still higher than the available resources, applying a penalty function.

The literature has implemented many penalty functions where the most fundamental one is the death penalty, eliminating any infeasible solution from consideration immediately. Another method is based on a constant magnitude penalty depending on the number of violated constraints. A more efficient method extracted from (Damak, Jarbouia, Siarryb and Loukila,2009) will be used in this study. This method measures the degree of the infeasibility of solutions from the feasible region and transforms it into a penalty that grows in proportion to the infeasibility level; thus, it is continuously guiding the search towards the feasible space. The penalty function is defined with Equations 1 and 2:

$$\text{Penalty} = \delta\,(C_{max})\,.\,\text{degree of infeasibility} \qquad \text{Eq. 1}$$

$$\text{Penalty} = \delta\,(C_{max}).\max\left(\frac{\text{Maximum of Daily Consumption}}{\text{Resource threshold level}} - 1, 0\right) \qquad \text{Eq.2}$$

Where, δ is the fitness factor of the current solution that takes a value larger than one and, C_{max} is

the fitness value of the current solution before adding the penalty. Accordingly, the precise definition of the fitness function for infeasible solutions is defined with Equation 3:

$$\text{Fitness value} = C_{max} + \text{penalty} \quad \text{Eq. 3}$$

Increasing the δ value will make the penalty a death penalty where the current solution will not survive in the next generation. On the other hand, decreasing the δ value will make the current population survive for the next generation. Accordingly, the trade-off must be found since we want to eliminate the population's infeasibility to find the optimal global solution and avoid stagnation in a local optimum by introducing some perturbations. When the penalty value is zero, it reveals that the current solution is feasible.

The objective of the developed optimization methods is to find a set of non-dominated solutions that optimize the project's total duration and cost based on activities execution modes with resources and precedence relationship constraints. As well to strike a balance between the resources acquiring and releasing costs and activity splitting costs. In multi-objective problems, there is always a conflict in the problem criteria. In other words, an improvement in one objective can only be achieved by sacrificing another. In addition, most criteria are non-commensurable; it is hard to predefine preferences among them before a search process is carried out. Accordingly, the best solution does not usually exist in such optimization problems. Instead, there usually exists a set of solutions, which are called non-dominated solutions or Pareto-optimal solutions. The developed method supports two optimization methods; elitist non-dominated sorting genetic algorithm (NSGA-II) and weighted multi-objective optimization genetic algorithm.

3.1.5.1 NSGA-II Optimization Method

NSGA algorithm was first suggested by Goldberg (1989), where the author presented the Pareto ranking process to carry the multi-objective optimization. This process, known as non-dominated sorting, is the cornerstone of all multi-objective evolutionary optimization algorithms (Zahraie and Tavakolan, 2009). NSGA utilizes fitness sharing to measure effective objectiveness based on the number of members occupying the same space or situation. Srinivas and Deb (1994) implemented NSGA for the first time in 1994. Later on, the NSGA method was criticized for its complexity algorithm in sorting nondominated points and ignoring elitism. Deb, Pratap, Agarwal and Meyarivan (2002) attempted to improve the NSGA algorithm by proposing NSGA-II. This fast and elitist multi-objective model proved to be more efficient than other previously developed algorithms. Unlike the traditional genetic algorithm, NSGA-II utilizes two extra operators: the fast-non-dominated sorting operator and fast crowded distance operator. NSGA-II utilizes these operators in the selection process to rank each individual in the population based on the multi-objective functions. On the other hand, the traditional genetic algorithm uses a simple selection operator based on selecting the lower fitness function values in the minimization problems. Yandamuri, Srinivasan and Bhallamudi (2006) introduced the following steps for the NSGA-II algorithm, which will be used in this study.

1- Selection of GA parameters, where the parameter setting may affect the solution quality and speed.
    - Set the number of generations as a termination condition, size of population N, and the probability of mutation and crossover.
2- Create a parent population (P0) randomly with a size N.
    - The chromosome representation contains the mode assignment for activities execution modes where each activity $j \in n$ is assigned a mode $m \in M_j$.

3- Based on the nondomination concept, Sort the present random population.

4- Assign a fitness rank value for each non-dominated solution equal to its nondomination level, where 1 is the best level, 2 is the next best level, and so on.

5- Divide the parent population into P_1 and P_2 to create a child population (Q_0) of size N using crossover and mutation operators.

6- Create a new generation from the initial population with the following steps:

- create R_t with a size of 2N (the mating pool) by combining the parent population (P_t) and the child population (Q_t).
- Identify all non-dominated fronts (F_1, F_2,…., F_l) using the fast nondominated sorting procedure to sort R_t.
- Generate the new parent population (P_{t+1}) of size N by adding nondominated solutions starting from the first ranked nondominated front and proceeding with the subsequently ranked nondominated fronts until exceeding size N.
- Reject some of the lower-ranked nondominated solutions from the last front to construct the new parent population P_{t+1} of size N using crowding distance sorting assigned to each solution in the previously dominated front.
- Create the new child population Q_{t+1} of size N by using the crossover and mutation operators on P_{t+1}

7- Repeat Step 6 until reaching the maximum number of generations.

8- Generate a 2D Pareto front for the output solutions showing the optimal time-cost trade-off curve.

A sample code on the NSGA-II method is provided in Appendix I. The NSGA-II functions were adopted from a program code written by Khan (2017), NSGA-II fast non-dominated sorting function, sorting solution, and crowding distance calculation function.

3.1.5.2 Dynamic Weighted Optimization Genetic Algorithm Method

A dynamic weighted optimization method is constructed to solve the multi-objective problem with cost and time as the objective functions to validate the developed NSGA-II method and compare the results. Integrating multiple objective functions is a challenge since they have different units of analysis. Accordingly, there has been a variety of developed methods to solve this issue. Some studies choose one primary objective and change the others into constraints with a fixed range value (Lee, Madanat and Reger, 2016). Conversely, other studies base their analysis on the normalization of the model output to compare the multi-objective solutions (Torres-Machi *et al.,* 2018). The latter method will allow all the objectives to be explored simultaneously in the objective function. The multi-objective problem was transformed into a single objective problem using dynamic weights, as shown in Equation 4. This method was used in the literature by Elmasry, Zayed and Hawari (2019) and France-Mensah and O'Brien (2019).

$$\text{Minimize } F(x) = \sum_{n=1}^{N} w_n f_n(x) \qquad \text{Eq.4}$$

Such that $\sum_{n=1}^{N} w_n = 1$ and N = total number of objective functions

To avoid any biases in choosing the objective functions' weights and to have an objective method, values were randomly generated to determine the optimal combination weights for the two objective functions (i.e., time and cost), as shown in Equation 5. These weights were considered default values; however, the user can input the relative weights that suit their project and conditions.

$$w_i = \frac{\mu_i}{\sum_{i=1}^{n} \mu_i} \quad \text{Eq. 5}$$

Where, μ_i is the importance of objective function i and $\sum_{i=1}^{n} \mu_i$ is the summation of all objective functions' importance.

Accordingly, the resultant single objective function is shown in Equation 6.

$$F(x) = w_1 * \text{Duration} + w_2 * \text{project cost} \quad \text{Eq. 6}$$

The developed code for the dynamic weighted optimization genetic algorithm method utilized a python library called GA.

3.1.6 Decision-Support Module

Multi-criteria decision-support methods permit considering different attributes for ranking the different scenarios and choosing the best one among the generated ones (Abdelkader, Marzouk and Zayed, 2019). The developed decision-support module uses the weighed sum method to allow the project managers to select the most feasible scenario among the generated ones. It generates a 2D Pareto fronts for time, cost solutions for the desired alpha-cut values, and project parameters where the optimum solution is selected. The weighted sum method is based on calculating a preference index for each alternative, whereas the best alternative is the one with the highest preference in the maximization case. On the other hand, the best alternative has the lowest preference in the minimization case.

The preference of each alternative can be calculated using Equation 7.

$$p_i = \sum_{j=1}^{n} f_{ij} * w_j (1 \leq i \leq m, 1 \leq j \leq n), \quad \text{Eq. 7}$$

Where pi represents the preference of each alternative. f_{ij} represents the measure of performance in the normalized matrix. w_j represents the weight of each criterion. m and n represent the number of alternatives and the number of criteria, respectively.

Accordingly, the normalized objective function for the proposed optimization method can be expressed with Equation 8.

$$\text{Normalized Objective function} = \left(\frac{\text{duration} * \text{weight}}{\text{Sum of durations}}\right) + \left(\frac{\text{cost} * \text{weight}}{\text{Sum of costs}}\right) \quad \text{Eq. 8}$$

3.2 Mathematical Formulation

Considering a construction project with n activities where each activity j has m modes of execution M_j = 1,2,…,m. The activity execution modes have fuzzy durations and costs based on the user input T_j and DC_j respectively, j = 1, 2,…,n, and a fixed resource requirement as a renewable resource p. The longest duration of an activity j is known as the normal duration, and the shortest duration is knows as the crash duration. The critical path calculations (forward and backward calculations) have resulted in NN noncritical activities and NC critical activities, forming a critical path(s) of a project duration T. Also, the earliest start time ES_j, earliest finish time EF_j, latest start time LS_j, Latest finish time LF_j, and total float TF_j of each activity has been calculated based on the user input precedence relationship. Each noncritical activity can be rescheduled within its early start and late finish and split based on the user input resource constraint. The resource constraint guarantees that the resource demand R_{tp} is less than or equal to the resource availability R_a for each project period.

This study developed an integrated method that optimizes project duration and cost through the resources and cost of the execution modes assigned to project activities. It accounts for project cost and resource-leveling based on costs and resources imbedded in these modes of execution.

The method's objective is to minimize the project duration and cost, including direct cost, indirect cost, and delay penalty, and strike a balance between the cost of acquiring and releasing resources on the one hand and the cost of activity splitting on the other hand. The objective functions can be represented with Equations 9 and 10.

$$\text{Minimize } T = \sum_{j=1}^{NC} \sum_{m=1}^{M_j} x_{jm} D_{jm}; \quad \text{Eq. 9}$$

$$\text{Minimize } PC = \sum_{p=1}^{P} [CA_p \sum_{t=1}^{T} I_{tp} + CR_p \sum_{t=1}^{T} D_{tp}] +$$

$$\sum_{j=1}^{N} [CS_j NL_j] + \sum_{j=1}^{N} \sum_{m=1}^{M_j} [x_{jm} C_{jm}] + IC\ T + [(T - F)B]; \quad \text{Eq. 10}$$

The decision variables of the developed method:

- x_{jm}= binary variable equals to 1 when activity j is performed under mode m and zero otherwise
- y_{tj}= binary variable equals to one when activity j is progressing at time t; t= ES_j *to* LF_j
- z_{jt}= Required resources by activity j during time t.
- NL_j= number of times activity j is split.
- D_{tp}= number of resources p released during time t
- I_{tp}= number of resources p acquired during time t
- S_j = start time of activity j; j=1…,nc
- F_j = finish time of activity j; j=1…,nc

The following notations will be used throughout this section.

- N = number of activities
- NC = number of critical activities
- M_j = number of modes for executing activity j; j=1,2,…….N
  under mode m and zero otherwise
- P = number of resource types required by the project
- D_{jm}= duration of activity j running in mode m; m=1,2…. M_j
  time t; t= ES_j *to* LF_j

- T = actual project duration
- B = Delay penalty/ Opportunity cost
- PC = project cost
- F = contract project duration
- CA_p = Acquiring cost of resource p
- CR_p = Releasing cost of resource p
- CS_j = Splitting cost of activity j
- IC = indirect cost
- C_{jm} = Direct cost of activity j under mode m

The method is developed under the following assumptions:

- Each activity has a constant resource requirement rate over its duration.
- All non-critical activities can be split with an associated cost.
- An activity resumes after splitting with the same resource requirement.
- Every activity has multiple modes but can be executed under one mode during its duration.
- The precedence relationship for split activities remains unchanged.
- The project resources are assumed to be interchangeable for the project activities.

3.3 Problem Constraints

The introduced decision variables should satisfy the following types of constraints: duration constraint, network logic constraint, resource balance constraint, activity splitting constraint. The mathematical formulations of these constraints are adopted from (Hariga, Shamayleh and El-Wehedi, 2016).

3.3.1 Duration Constraint

This constraint states that the total number of active periods for a non-critical activity j should be equal to it its duration D_{jm} $when\ running\ under\ mode\ m$ as seen in Equation 11.

$$\sum_{t=ES_j}^{LF_j} y_{tj} = \sum_{m=1}^{M_j} D_{jm} x_{jm} \; ; j = 1, \ldots, nn \qquad \text{Eq. 11}$$

3.3.2 Network Logic Constraint

The network logic constraint guarantees that the precedence relationships between all activities are preserved. To express these network logic constraints, we need first to determine each activity's mode. Equation 12 represents mathematically that each activity is only allowed to run in one mode of operation.

$$\sum_{m=1}^{M_j} x_{jm} = 1; \; j=1,\ldots., N \qquad \text{Eq. 12}$$

Furthermore, we will also need to determine the starting and finishing times for all activities using Equations 13 and 14.

$$S_j = (T+1) - \max\{(T+1-t)y_{tj}; t = ES_j, ES_j + TF_j\} \qquad \text{Eq. 13}$$

$$F_j = \max\,(\, t\, y_{tj}; t = LF_j - TF_j, LF_j) \qquad \text{Eq. 14}$$

Thus, the network logic constraint can be represented by Equation 15.

$$S_k \geq F_j + 1; \; j=1,\ldots., N \; k \in succ\,(j) \qquad \text{Eq. 15}$$

where succ(j) is the set of immediate successors to activity j.

3.3.3 Resource Balance Constraint

The resource balance constraint ensures that the resource requirement for resource p on period t plus the amount of the same resource released during period t is equal to the resource requirement on period t-1 plus the number of resources acquired during period t. Before defining this constraint, we need to express the resource requirement (R_{tp}) as a function in terms of the

binary variable y_{tj} and x_{jm}. In other words, the resource requirement at time t is the sum of the required resources for all activities running at time t, which can be explained with Equation 16.

$$R_{tp} = \sum_{j=1}^{N} y_{tj} \sum_{m=1}^{M_j} r_{jm} x_{jm}; t = 1, \ldots\ldots, T \quad \text{Eq. 16}$$

Since the above function is nonlinear as it involves the product of the decision variables y_{tj}, and x_{jm}, we needed to convert it into a linear equation by introducing the continuous decision variable z_{jt} that is shown in Equation 17 and satisfying the constraints shown in Equations 18 to 20:

$$z_{jt} \leq r_j^{max} y_{tj}; j=1,\ldots\ldots,N \quad t= ES_j \text{ to } LF_j \quad \text{Eq. 17}$$

$$z_{jt} \leq \sum_{m=1}^{M_j} r_{jm} x_{jm} ; j=1,\ldots\ldots,N \; t= ES_j \text{ to } LF_j \quad \text{Eq. 18}$$

$$z_{jt} \geq \sum_{m=1}^{M_j} r_{jm} x_{jm} - r_j^{max}(1 - y_{tj}) ; j=1,\ldots\ldots,N, \; t= ES_j \text{ to } LF_j \quad \text{Eq. 19}$$

$$z_{jt} \geq 0 ; j=1,\ldots\ldots,N, \; t= ES_j \text{ to } LF_j \quad \text{Eq. 20}$$

Using the above constraints, it can be verified that $z_{jt}= 0$ when $y_{tj}= 0$. Otherwise,

$$z_{jt}= \sum_{m=1}^{M_j} r_{jm} x_{jm} \quad \text{when } y_{tj}=1 \quad \text{Eq. 21}$$

By using the new continuous decision variables, the resource requirement can be written with Equation 22.

$$R_{tp} = \sum_{j=1}^{N} z_{jtp} ; t = 1, \ldots\ldots, T \; p = 1, \ldots.., P \quad \text{Eq. 22}$$

The resource balance constraint can be written, as shown in Equation 23.

$$R_{tp} - R_{(t-1)p} + D_{tp} - I_{tp} = 0 \quad \text{Eq. 23}$$

3.3.4 Activity Splitting Constraint

Activity splitting constraint helps determine the number of times an activity has been split during its scheduling interval as a function of the binary variable y_{tj}. An activity is considered split at period t+1when $y_{tj} = 1$ and $y_{(t+1)j} = 0$ for $t \geq ES_j\, j$ and $t < LF_j - 1$.

This condition can be presented mathematically with Equation 24:

$$L_{tj} = \max\{y_{tj} - y_{(t-1)j}, 0\}\ j = 1, \ldots, N,\ t = ES_j \text{ to } LF_j \qquad \text{Eq. 24}$$

This equation specifies that $L_{tj} = 1$ when $y_{tj} = 1$ *and* $y_{(t+1)j} = 0$, indicating a split occurred during the period (t + 1). Then, the number of times activity j has been split during its scheduling interval can be easily determined using Equation 25:

$$NL_j = \sum_{t=ES_j}^{LF_j} L_{tj} - 1;\ j = 1, \ldots, N \qquad \text{Eq. 25}$$

Note that when the finish time of activity j is less than LF_j , then using the above Equation L_{tj} =1 for $t = F_j$ will result in zero, which confirms that it is not a splitting case during the period F_j +1. This case explains the use of (−1) in the equation.

4 CHAPTER FOUR: NUMERICAL EXAMPLE

4.1 Background

The developed method is applied to a numerical example adapted from (Zheng and Ng, 2005) to demonstrate its use and capabilities. This numerical example was utilized since it was widely used in the project scheduling under uncertainty studies and covered much of the parameters used in this study. The example project network is shown in Figure 11.

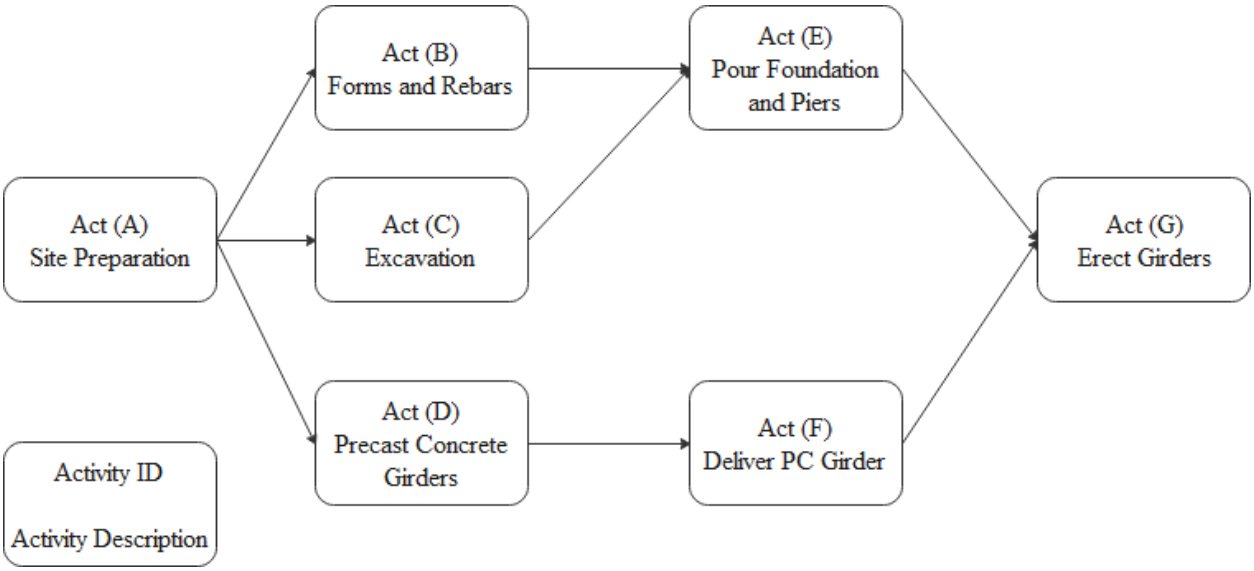


Figure 11 Numerical example network (Zheng and Ng, 2005)

The project consists of seven activities (A-G) where each activity has three to five modes of execution to be selected from, as shown in Tables 3 and 4. Triangle fuzzy numbers represent the time and cost of each option. The first and third values for time and cost represent the lowest and highest possible values for each option. The second value defines the most probable time and costs for that option. The sum method was used as a decision-support module to find the best time-cost solution from the generated set of non-dominated solutions.

Table 3: Activity execution modes (costs) (Zheng and Ng, 2005)

	Direct cost in $ 1000 (optimistic, most likely, pessimistic)				
Activity	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
A	(20.5, 23, 28)	(16.2, 18, 21.2)	(11.5, 12, 13.5)	-	-
B	(2.87, 3, 3.58)	(2.26, 2.4, 2.9)	(1.5, 1.8, 2.1)	(1.08, 1.2, 1.9)	(0.5, 0.6, 0.72)
C	(4.2, 4.5, 4.95)	(3.75, 4, 4.5)	(2.9, 3.2, 3.75)	-	-
D	(43.5, 45, 48.85)	(33.62, 35, 38.9)	(28.5, 30, 34.5)	-	-
E	(18.1, 20, 23.5)	(15.2, 17.5, 21)	(13.7, 15, 18.5)	(8.75, 10, 11.8)	-
F	(38.5, 40, 49.5)	30.25, 32, 36.2)	(16.8, 18, 21.05)	-	-
G	(28.4, 30, 34.5)	(22.2, 24, 26.8)	(21, 22, 23.25)	-	-

Table 4: Activity execution modes (durations) (Zheng and Ng, 2005)

	Duration in days (optimistic, most likely, pessimistic)				
Activity	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
A	(12, 14, 16)	(17, 20, 23)	(19, 24, 29)	-	-
B	(12, 15, 18)	(13, 18, 23)	(15, 20, 25)	(23, 30, 37)	(54, 60, 66)
C	(10, 15, 20)	(20, 22, 24)	(28, 33, 38)	-	-
D	(10, 12, 14)	(12, 16, 20)	(17, 20, 23)	-	-
E	(16, 22, 28)	(20, 24, 28)	(23, 28, 33)	(25, 30, 35)	-
F	(12, 14 16)	(15, 18, 21)	(20, 24, 28)	-	-
G	(7, 9, 11)	(13, 15, 17)	(15, 18, 21)	-	-

Unfortunately, since there is little research in the resource-constrained scheduling under uncertainty, some modifications were applied to the project parameters to perform the study with a resource-constrained environment.

The modifications are as follows:

1- Activity's needed resources were calculated using Equation 25 and presented in Table 5, where the resource cost was assumed to be 200$/day:

- Resources needed=Activity cost/ (activity duration * 200$/day) Eq. 25

Table 5: Activity execution modes (resources)

	Project Activities' Resources				
Activity	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
A	9	5	3	-	-
B	1	1	1	1	1
C	2	1	1	-	-
D	22	14	8	-	-
E	6	4	3	2	-
F	16	10	4	-	-
G	20	9	7	-	-

2- The used project parameters are shown in Table 6, where the splitting and resources acquiring and releasing costs were assumed based on values from the literature.

Table 6: Project parameters

Indirect cost ($/day)	500
Contractual duration (day)	80
Splitting cost ($/split)	1000
Acquiring costs ($/resource)	500
Releasing costs ($/resource)	500
Delay penalty/ Opportunity cost ($/day)	500
Available resources (resource/day)	15
Alpha-cut value (i.e. accepted level of risk)	0.5

4.2 Method Validation

The resource-constrained time-cost optimization problem was solved with the dynamic weighted optimization method and NSGA-II to evaluate their results and illustrate their robustness. The developed method was coded using python on a 16GB RAM, 2.60 GHz i7 core CPU, and Windows 10 with a 64-bit operating system. The running time ranged between 10 min and 15 min.

The proposed methods are validated through a multi-layered comparative analysis that involves performance evaluation and statistical comparisons. The used performance evaluation method compares the quality of the non-dominated set of solutions by ranking the solutions and finding the best alternative using the weighted sum method. On the contrary, the statistical comparison investigates the capabilities of the developed methods in improving the solutions by genetic algorithm evolution and searching the unknown criteria space before converging to a

global optimum. The used statistical measures in the validation process are the mean, range, and best values. The coefficient of variation was used to assess the stability of the developed optimization methods since this test is used widely in the literature for that purpose.

Genetic Algorithm Sensitivity analysis

A sensitivity analysis of the GA parameters is performed to identify the GA parameters that produce quality solutions. The developed sensitivity module is case-dependent, which is very useful in optimization problems where the knowledge about the optimal GA parameters is unknown. As our case study is rather simple, the algorithms may have a high tendency to converge impulsively at a local optimum. Accordingly, the recommended genetic algorithm parameters from the literature are as follows: population size = 50, the number of generations = 200, two parents exchanged their genes with a crossover rate of 0.4 to ensure adequate offspring exploitation, exploration, and stepwise convergence, a mutation rate of 0.9 which aids in avoiding premature convergence and introducing useful genetic materials. Sensitivity analysis is performed for different values of crossover probabilities (P_c = 0.4 to 1 in steps of 0.2), mutation probabilities (P_m = 0.2 to 0.8 in steps of 0.2), and population sizes (30, 50, and 60). The values of the parameters were changed iteratively to test the effect on the quality of solutions.

Table 7 and Figure 12 show the sensitivity analysis results for the population size. As can be seen, the different population sizes did not show a big difference in the output solutions and the optimal time-cost trade-off curves. However, the 2nd solution in the recommended population size had the lowest preference index (PI) among all other solutions of the different population sizes with a PI value of 0.080252, a duration of 61 days, and a cost of $183,781. While the best solution for population sizes 30 and 60 had PI values of 0.080281 and 0.080294, durations of 61 days and

61 days, and costs of $183,907 and $183,965, respectively. Accordingly, having a genetic algorithm run with a 50-population size will result in better solutions.

Table 7: Population size sensitivity analysis

	Duration (Days)	Cost ($)	Normalized Duration	Normalized Cost	PI
Pop size = 30	60	213147	0.075472	0.09716	0.086316
	61	**183907**	**0.07673**	**0.083832**	**0.080281**
	67	173320	0.084277	0.079006	0.081641
	73	167396	0.091824	0.076305	0.084065
	60	197889	0.075472	0.090205	0.082838
Recommended pop size = 50	60	211004	0.075472	0.096184	0.085828
	61	**183781**	**0.07673**	**0.083774**	**0.080252**
	73	167522	0.091824	0.076363	0.084093
Pop size = 60	**61**	**183965**	**0.07673**	**0.083858**	**0.080294**
	67	176478	0.084277	0.080445	0.082361
	73	170304	0.091824	0.077631	0.084727
	79	165050	0.099371	0.075236	0.087304
	60	211004	0.075472	0.096184	0.085828

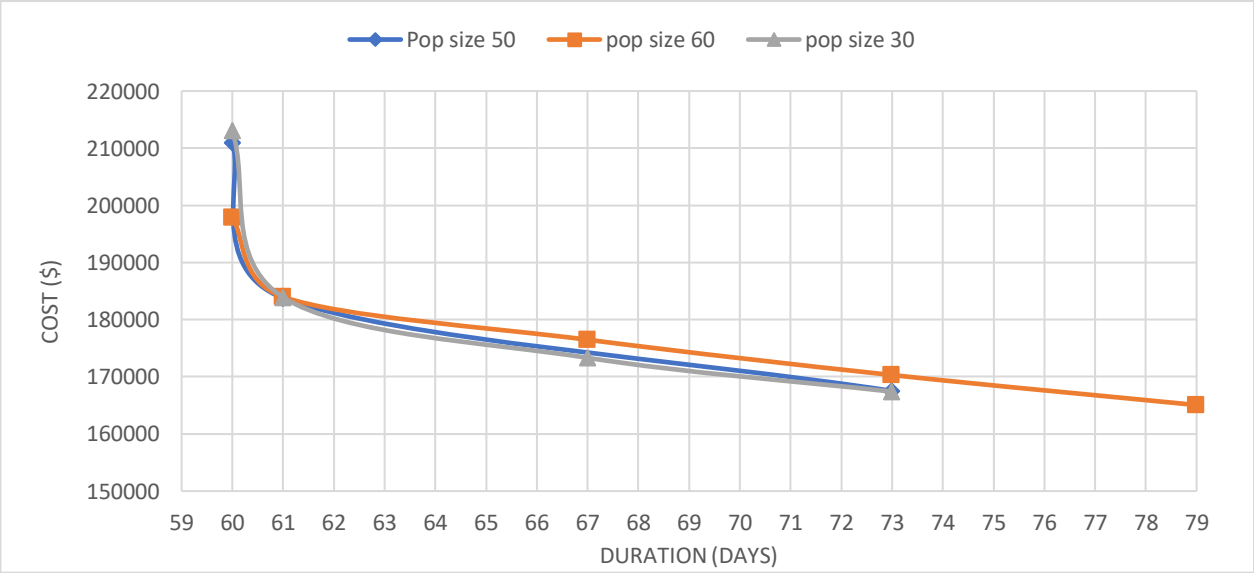


Figure 12 Population size sensitivity analysis

Table 8 and Figure 13 illustrate the sensitivity analysis results for the mutation rate. It can be observed that mutation rates of 0.4 and 0.6 resulted in higher PI solutions with values of 0.068147 and 0.068146, respectively, and better time-cost trade-off curves than that of the mutation rates of 1 and 0.8 with PI values of 0.070773 and 0.069212 respectively. Although mutation rates of 0.4 and 0.6 resulted in very close PI values, the mutation rate of 0.6 had a better time-cost trade-off curve, as shown in Figure. Accordingly, choosing either of mutation rates of 0.4 and 0.6 should result in a better set of non-dominated solutions.

Table 8: Mutation rate sensitivity analysis

	Duration (Days)	**Cost ($)**	**Normalized Duration**	**Normalized Cost**	**PI**
Mutation rate = 0.4	60	197785	0.062827	0.077939	0.070383
	61	**183781**	**0.063874**	**0.072421**	**0.068147**
	76	165289	0.079581	0.065134	0.072357
Mutation rate = 0.6	60	197785	0.062827	0.077939	0.070383
	61	**183775**	**0.063874**	**0.072418**	**0.068146**
	67	170288	0.070157	0.067104	0.06863
	73	167407	0.07644	0.065968	0.071204
	82	165705	0.085864	0.065298	0.075581
Recommended mutation rate = 0.8	60	197785	0.062827	0.077939	0.070383
	67	**173239**	**0.070157**	**0.068266**	**0.069212**
	82	168163	0.085864	0.066266	0.076065
Mutation rate = 1	60	213170	0.062827	0.084002	0.073414
	67	**181164**	**0.070157**	**0.071389**	**0.070773**
	79	172354	0.082723	0.067918	0.07532

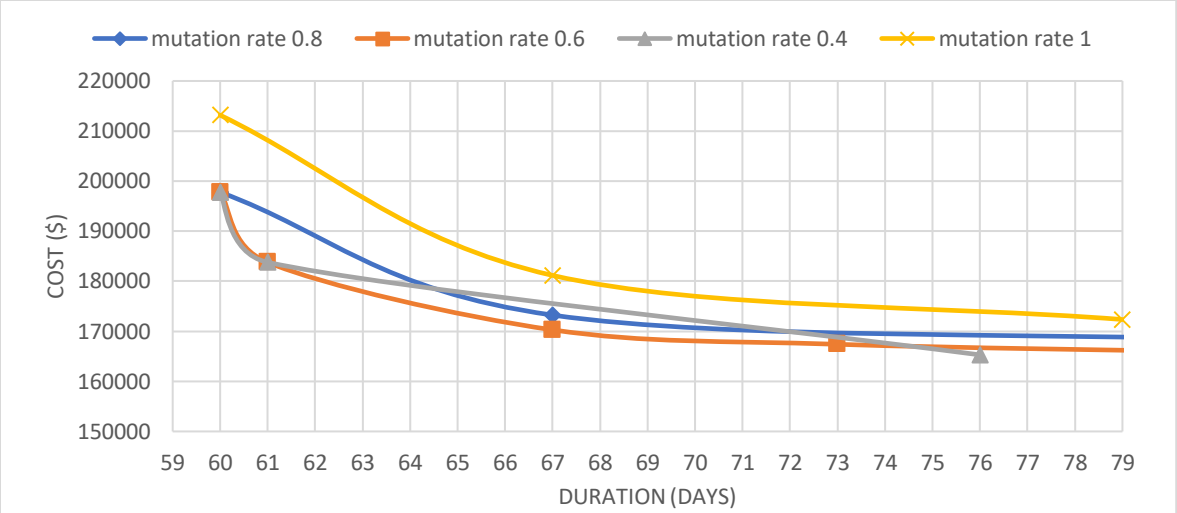


Figure 13 Mutation rate sensitivity analysis

Table 9 and Figure 14 demonstrate the sensitivity analysis results for the crossover rate. It can be seen that the output solutions and time-cost trade-off curves didn't follow any pattern for the different crossover rates. However, the crossover rate of 0.4 had the lowest PI value of 0.055726 with a duration of 61 days, and a cost of $183,781among all other solutions of the different crossover rates. While the best solution for mutation rates 0.2, 0.6, and 0.8 had PI values of 0.055733, 0.05576, and 0.057515, durations of 61 days, 61 days, and 60 days, and costs of $183,820, $183,995, and $197,666 respectively. Consequently, having a genetic algorithm run with a 0.4 crossover rate can result in better solutions.

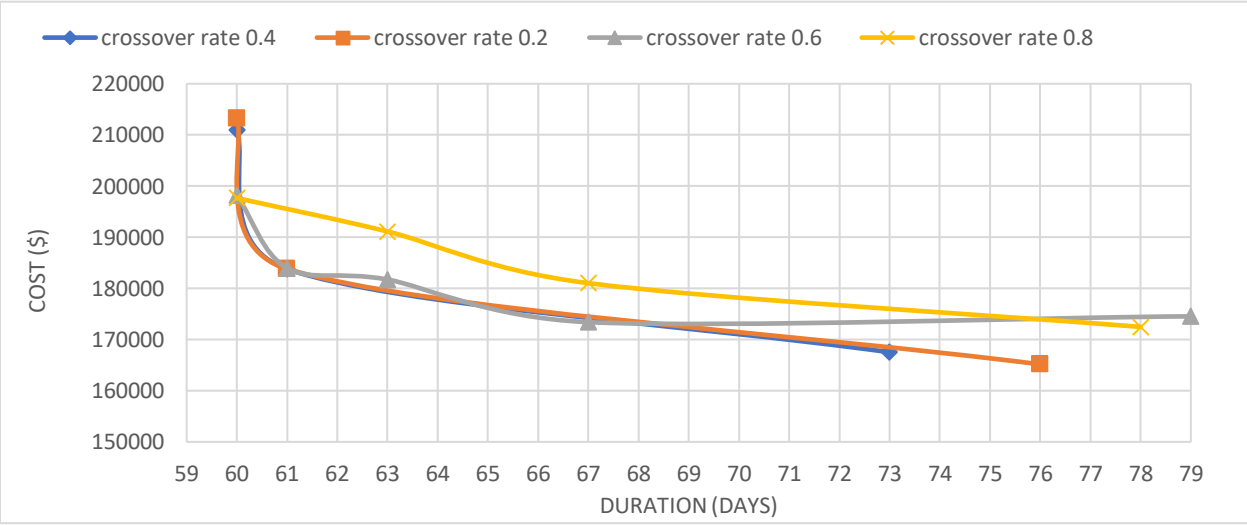


Figure 14 Crossover rate sensitivity analysis

Table 9: Crossover rate sensitivity analysis

	Duration (Days)	Cost ($)	Normalized Duration	Normalized Cost	PI
Crossover = 0.2	60	213219	0.051813	0.06819	0.060002
	61	**183820**	**0.052677**	**0.058788**	**0.055733**
	76	165157	0.06563	0.052819	0.059225
Recommended crossover = 0.4	60	211004	0.051813	0.067482	0.059648
	61	**183781**	**0.052677**	**0.058776**	**0.055726**
	73	167522	0.06304	0.053576	0.058308
Crossover = 0.6	60	198194	0.051813	0.063385	0.057599
	61	**183995**	**0.052677**	**0.058844**	**0.05576**
	63	181741	0.054404	0.058123	0.056264
	67	173403	0.057858	0.055456	0.056657
	79	174530	0.068221	0.055817	0.062019
	81	173628	0.069948	0.055528	0.062738
	88	174530	0.075993	0.055817	0.065905
Crossover = 0.8	**60**	**197666**	**0.051813**	**0.063216**	**0.057515**
	63	191131	0.054404	0.061126	0.057765
	67	181032	0.057858	0.057896	0.057877
	78	172477	0.067358	0.05516	0.061259

Based on the performed sensitivity analysis results, the NSGA-II and dynamic weighted genetic algorithm method will use the following genetic algorithm parameters: Population size = 50, number of generations = 200, mutation rate = 0.6, and crossover rate = 0.4.

5 CHAPTER FIVE: DISCUSSION OF RESULTS

5.1 NSGA-II Optimization Method Results

After running 200 generations of the NSGA-II optimization method using a population size of 50, a mutation rate of 0.6, and a crossover rate of 0.4, the non-dominated set of solutions is shown in Table 10, where the sum weighted method was used to find the best solution among them. Since our optimization problem is a minimization case, the lowest preference index (PI) solution will be the best alternative. It can be seen that the third solution is the best solution with the lowest PI with a value of 0.160327573, a duration of 69 days, and a cost of $178,536. Simultaneously, the sixth solution is the most inadequate solution with the highest PI value of 0.1796517, a duration of 90 days, and a cost of $169,459. Figure 15 shows the resulted solutions for the 200 generations, where the black curve shows the optimal time-cost trade-off curve with the non-dominated set of solutions. The selected mode for each activity based on the optimal solution is highlighted in Table 11.

Table 10: NSGA-II non-dominated set of solutions

Duration (Days)	Cost ($)	Normalized Duration	Normalized Cost	Preference Index (PI)
67	204142	0.150224215	0.189747031	0.169985623
68	181709	0.152466368	0.168895883	0.160681125
69	**178536**	**0.15470852**	**0.165946625**	**0.160327573**
73	171895	0.16367713	0.159773912	0.161725521
79	170123	0.177130045	0.158126864	0.167628454
90	169459	0.201793722	0.157509685	0.179651704

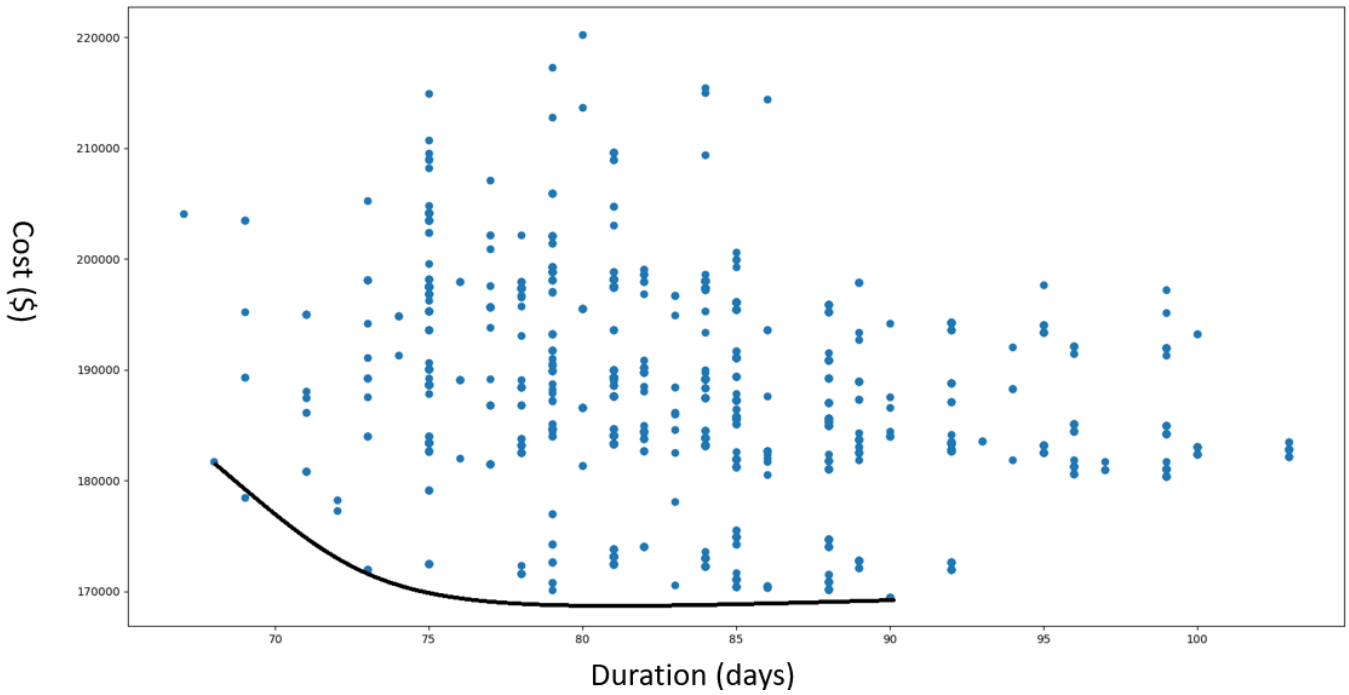


Figure 15 NSGA-II Pareto front

Table 11: Selected mode for each activity (NSGA-II)

	Project Activities Durations (days)				
Activity	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
A	✔	✕	✕	-	-
B	✔	✕	✕	✕	✕
C	✕	✔	✕	-	-
D	✕	✕	✔	-	-
E	✕	✔	✕	✕	-
F	✕	✔	✕	-	-
G	✔	✕	✕	-	-

NOTE: Correct sign = Elected mode, Cross = Mode available for the activity but not selected, Blank = Mode wasn't available for that activity

5.2 Dynamic Weighted Optimization Method Results

Since the dynamic weighted optimization method requires selecting weights for the two objective functions (i.e., duration and cost), 100 values were randomly generated to avoid any biases in choosing the objective functions' weights. A sample of the generated weighs shown in Table 12, where the non-dominated solutions of each weight's combination, are listed with their calculated weighted objective function. The optimal solution for each weight's combinations is determined based on the weighted sum method as the solution with the lowest weighted objective function. For example, the optimal solution for run 20 with W1= 0.7868, W2= 0.2132, is the solution with the lowest weighted objective function with a value of 36451.1, a duration of 86 days, and a cost of $170,667.5. Table 13 summarizes the optimal solutions for some of the chosen weight combinations. The best alternative among these optimal solutions, which is written in bold in Table 13, is the solution with the lowest preference index (PI) value. This solution has a PI value of 0.02421525, a duration of 73 days, a cost of $148,257, and objective function weights of 0.10438 (W1) and 0.895618 (W2). These weights indicate that cost was predominant in this optimization process.

Dynamic weighted optimization was run for 200 generations using the found best alternative weight combination, a population size of 50, a mutation rate of 0.6, and a crossover rate of 0.4. The resulting non-dominated set of solutions is shown in Table 14, where the sum weighted method was used to find the best solution. Since our optimization problem is a minimization case, the solution with the lowest preference index is the best alternative. The third solution has the lowest PI with a value of 0.126069221, a duration of 82 days, and a cost of $155,017. On the other hand, the sixth solution has the highest PI with a value of 0.1759102, a duration of 116 days, and

a cost of $212,947. The selected mode for each activity based on the optimal solution is highlighted in Table 15.

Table 12: Sample of the generated weights

Iterations	Objective functions weights	Duration (Days)	Cost ($)	Weighted objective function
1	W1= 0.3795731 and W2 = 0.6204268	73	171700	106555
		79	250264.7	155300.9
		70	255179.6	158346.8
		67	**161487.5**	**100216.6**
		75	264603.8	164195.7
		94	292847.7	181726.2
		73	444113.7	275567.7
		61	258210.4	160223.8
20	W1 = 0.7868169 and W2 = 0.2131830	119	203370	43448.7
		86	244215.7	52130.3
		127	195000	41670.6
		86	**170667.5**	**36451.1**
		83	261197.5	55748.2
		91	309955	66148.7
		60	306686	65427.5
		92	210567.3	44961.8
69	W1= 0.1043815 and W2 = 0.8956184	**73**	**148257.5**	**132789.8**
		83	286221.9	256354.3
		68	277004.2	248097.1
		73	171587.5	153684.5
		85	287024.8	257073.6
		77	413033.8	369928.7
		68	277004.1	248097.1
		93	283374.4	253805.1
100	W1 = 0.2274323 and W2 = 0.7725676	**77**	**160262.5**	**123831.1**
		84	265201.4	204905.1
		92	278500	215181
		91	168495	130194.5
		117	424215.3	327761.6
		122	193135	149237.6
		83	285333.6	220458.4
		76	268445.2	207409.3

Note: The bold values show the optimal solution for each of the weights' combination.

Table 13: Optimal solutions for some of the chosen weight's combinations

Iteration	W1	W2	Duration (Days)	Cost ($)	Weighted objective function	Preference index
1	0.379573	0.620427	67	161487	100216	0.025909185
2	0.053117	0.946883	100	166982.5	158118.3	0.027375094
3	0.32203	0.67797	90	215718	146279	0.034661388
4	0.666265	0.333735	69	803035	268046.7	0.060746401
5	0.165209	0.834791	95	180300	150528	0.030187488
6	0.288942	0.711058	79	184754	131394	0.02989024
7	0.095044	0.904956	85	165265	149565.6	0.027079133
8	0.321706	0.678294	82	163850	111164	0.028029957
9	0.064577	0.935423	84	187637	175525.9	0.030205733
10	0.393901	0.606099	79	247652.1	150132.8	0.036106663
11	0.572025	0.427975	84	234665	100479	0.034724733
12	0.29564	0.70436	83	160495	113070	0.027620701
13	0.208417	0.791583	124	178982	141705.4	0.032711451
14	0.93771	0.06229	88	186887	11723	0.033884897
15	0.990365	0.009635	105	208247	2110	0.04067371
16	0.435746	0.564254	84	166615	94049	0.029255123
17	0.386916	0.613084	81	177165	108648	0.029551112
18	0.708912	0.291088	96	177487	51732	0.034680296
19	0.334022	0.665978	87	176570	117620	0.030103104
20	0.786817	0.213183	86	170667	36451.08	0.032082739
21	0.055499	0.944501	73	155970	147317.9	0.025157119
22	0.50942	0.49058	92	262577.9	128862.4	0.038809894
23	0.434954	0.565046	79	155025	87630.63	0.02735804
24	0.160908	0.839092	89	186005	156089.6	0.030544852
25	0.726364	0.273636	87	164357.5	45037.26	0.031722553
26	0.195595	0.804405	86	241866.7	194575.7	0.037676319
27	0.366043	0.633957	73	272249.2	172620.9	0.038001458
–	–	–	–	–	–	–
69	**0.10438**	**0.895618**	**73**	**148257**	**132789**	**0.02421525**
–	–	–	–	–	–	–
100	0.227432	0.772568	77	160262	123831.1	0.026618134

Table 14: Weighted objective optimization non-dominated set of solutions

Duration (Days)	Cost ($)	Normalized Duration	Normalized Cost	Preference Index (PI)
88	184377.5	0.143089431	0.141306555	0.142197993
76	179702.5	0.123577236	0.137723645	0.13065044
82	**155017.5**	**0.133333333**	**0.118805109**	**0.126069221**
87	195520	0.141463415	0.149846146	0.14565478
84	190537.5	0.136585366	0.146027567	0.141306467
116	212947.5	0.188617886	0.163202548	0.175910217
82	186702.5	0.133333333	0.143088431	0.138210882

Table 15: Selected mode for each activity (weighted objective optimization)

	Project Activities Durations (days)				
Activity	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
A	✔	✕	✕	-	-
B	✕	✕	✕	✔	
C	✔	✕	✕	-	-
D	✕	✕	✔	-	-
E	✕	✕	✕	✔	-
F	✕	✕	✔	-	-
G	✔	✕	✕	-	-

5.3 Comparison of the Developed Optimization Methods

Both of the developed optimization Methods' results were compared to evaluate their performance in finding the optimal solutions. The results of both methods are tabulated in Table 16, where it's evident that the NSGA-II was superior in finding better solutions than the dynamic weighted optimization. The optimal solution from the NSGA-II Pareto front has a PI value of 0.070013517, a duration of 69 days, and a cost of $178,536, while the optimal solution of the dynamic weighted optimization has a PI value of 0.07120034, a duration of 82, and a cost of $155,017.

Table 16: Comparison of the optimization methods results

Optimization Methods	Duration (Days)	Cost ($)	Normalized Duration	Normalized Cost	Preference Index
NSGA-II	67	204142	0.063147974	0.085749846	0.07444891
	68	181709	0.064090481	0.076326864	0.070208673
	69	**178536**	**0.065032988**	**0.074994046**	**0.070013517**
	73	171895	0.068803016	0.072204494	0.070503755
	79	170123	0.074458058	0.071460165	0.072959112
	90	169459	0.084825636	0.071181252	0.078003444
Dynamic weighted optimization	88	184377.5	0.082940622	0.077447768	0.080194195
	76	179702.5	0.071630537	0.075484034	0.073557286
	82	**155017.5**	**0.07728558**	**0.0651151**	**0.07120034**
	87	195520	0.081998115	0.082128175	0.082063145
	84	190537.5	0.079170594	0.080035276	0.079602935
	116	212947.5	0.10933082	0.089448596	0.099389708
	82	186702.5	0.07728558	0.078424384	0.077854982

On the contrary, the statistical test results were compared to investigate the developed methods' capabilities in improving the solutions by genetic algorithm evolution and searching the unknown criteria space before converging to a global optimum. The achieved statistical results

from both optimization methods are shown in Table 17. Both optimization methods solved the multi-objective time cost optimization problem since the objective functions (time and cost) were improved simultaneously with evolution. For example, the best duration and cost values for the NSGA-II method were improved from 103 days in generation 25 to 96 days in generation 200 and from $204,170 in generation 25 to $198,007 in generation 200. The best duration and cost values for the dynamic weighted optimization were improved from 78 days in generation 25 to 67 days in generation 200 and from $170552.5 in generation 25 to $155,017 in generation 200. However, NSGA-II appears more robust in searching out the optimal solutions since the NSGA-II output average values for duration and cost were lower than that of the dynamic weighted optimization. The NSGA-II average values in generation 200 for duration and cost are 82.94 days and $189,444.45, respectively. In comparison, the dynamic weighted optimization average values in generation 200 for duration and cost are 85.25 days and $185,500.93, respectively. Furthermore, the range values demonstrate that NSGA-II has a more remarkable ability in searching the unknown criteria space before converging to a global optimum. This explains the reason behind the high range values for the duration and cost in the first 25 generations of the NSGA-II compared to that of the last generations. Hence, the NSGA-II had a trade-off between exploration (i.e., exploring the new search space) and exploitation (i.e., using already detected points to search the optimum). On the other hand, the range values for the weighted dynamic optimization's duration and cost increased with the generations, which suggests that the algorithm didn’t have a trade-off between exploration and exploitation.

Table 17: Statistical analysis results

Optimization Methods	Iterations	Criteria					
		Duration (Days)			Cost ($)		
		Average	Best	Range	Average	Best	Range
NSGA-II	25	85.82	103	28	192210.1	204170	33817.5
	50	80.56	84	11	190923.0	198120	14075.0
	75	81.7	88	13	190610.9	201392.5	28415.0
	100	80.28	92	6	197758.8	198195.0	8880.0
	125	81.38	92	10	195475.3	198195.0	25030.0
	150	81.42	96	17	192486.8	198195.0	16250.0
	175	84.44	99	21	187402.05	204170.0	33970.0
	200	82.94	96	10	189444.45	198007.5	14150.0
Dynamic weighted optimization	25	87.375	78	38	189670.93	170552.5	42395.0
	50	86.625	78	38	188554.06	163995.0	48952.5
	75	86.625	78	38	186247.81	163995.0	48952.5
	100	86.75	75	41	188010.93	163995.0	48952.5
	125	86.25	75	41	187954.06	163995.0	48952.5
	150	84.5	73	43	184591.56	156437.5	56510.0
	175	85.625	73	43	188777.5	155017.5	57930.0
	200	85.25	67	49	185500.93	155017.5	57930.0

The coefficient of variation (CV), which is the ratio of the standard deviation to the mean, is a way to measure the optimization algorithm's stability. A CV closer to zero in the last generations is desirable as it shows little or no variation between the output solutions and reveals the algorithm stability. Table 18 shows the CV results of duration and cost for the two optimization methods through the different generations. It is shown that the NSGA-II had lower CV values for the duration and cost than the dynamic weighted optimization. NSGA-II had a CV value of 0.04678 and 0.05490 in the last generation for the duration and cost, respectively. However, the dynamic weighted optimization had a CV value of 0.15495 and 0.082734 in the last generation for the duration and cost, respectively. Accordingly, the results show that the NSGA-II performance was more stable than that of the dynamic weighted genetic algorithm.

Table 18: Coefficient of variation results

Optimization Methods	Iterations	Criteria					
		Duration (Days)			Cost ($)		
		Mean	Standard Deviation	Coefficient of variation	Mean	Standard Deviation	Coefficient of variation
NSGA-II	25	85.82	7.3	0.085062	192210.1	9432.85	0.049076
	50	80.56	2.041	0.025335	190923	6491.97	0.034003
	75	81.7	3.38	0.041371	190610.9	6381.76	0.033481
	100	80.28	1.94	0.024165	197758.8	1737.4	0.008785
	125	81.38	2.87	0.035267	195475.3	4858.6	0.024855
	150	81.42	3.42	0.042004	192486.8	5937.36	0.030846
	175	84.44	5.68	0.067267	187402.1	8074.37	0.043086
	200	82.94	3.88	0.046781	189444.5	10401.68	0.054906
Dynamic weighted optimization	25	87.375	11.21	0.128298	189670.93	11093.83	0.05849
	50	86.625	11.34	0.130909	188554.06	12618.67	0.066923
	75	86.625	11.34	0.130909	186247.81	13983.13	0.075078
	100	86.75	12.19	0.140519	188010.93	13072.67	0.069531
	125	86.25	11.66	0.135188	187954.06	12783.87	0.068016
	150	84.5	12.77	0.151124	184591.56	16066.68	0.087039
	175	85.625	12.09	0.141197	188777.5	15667.89	0.082997
	200	85.25	13.21	0.154956	185500.93	15347.23	0.082734

6 CHAPTER SIX: SUMMARY AND CONCLUSIONS

This study introduced an integrated method for the resource-constrained schedule compression under uncertainty that handles resource planning and project scheduling. The developed method was developed in a computational framework coded in python as a stand-alone automated computerized tool to aid in the iterative rescheduling of project activities and facilitate the project schedule optimization. The developed method evaluated two different genetic algorithm methods in the optimization process; NSGA-II and weighted multi-objective optimization. The two optimization methods went through a multi-layered comparative analysis to evaluate their performance and compare their outputs. Results showed that NSGA-II outperformed the weighted optimization method, resulting in a better global optimum solution and avoiding entrapment in local minima. It is anticipated that the developed method can help contractors generate efficient schedule compression while ensuring efficient utilization of resources.

The contributions of the developed method are in the field of project scheduling, specifically:

1- developing a multi-objective optimization method under uncertainty that tackles the time-cost trade-off problem and allows for activity splitting while providing a smooth resource utilization;
2- generating (2D) Pareto front for the duration and cost to allow project managers to select the optimum solution from a set of non-dominated solutions based on the accepted risk levels; and
3- computerizing the developed method in a stand-alone automated tool that can be used by contractors for schedule compression.

The developed method can be further extended to account for linear projects such as pipeline installations, highway projects, and multi-story building construction. As well, the developed algorithm can be further extended to consider the different types of resources. Finally, the developed automated computer application can be enhanced with a graphical user interface to facilitate its use.

References

Abdel-Basset, M., Ali, M., & Atef, A. (2019). Resource levelling problem in construction projects under neutrosophic environment. *The Journal of Supercomputing*, *76*(2), 964–988. doi: 10.1007/s11227-019-03055-6

Abdelkader, E. M., Marzouk, M., & Zayed, T. (2019). An optimization-based methodology for the definition of amplitude thresholds of the ground penetrating radar. Soft Computing, 23(22), 12063-12086. doi:10.1007/s00500-019-03764-3

Afruzi, E. N., Aghaie, A., & Najafi, A. A. (2018). Robust Optimization for the Resource Constrained Multi-Project Scheduling Problem with Uncertain Activity Durations. *Scientia Iranica*, *0*(0), 0–0. doi: 10.24200/sci.2018.20801

Afshar, A., Ziaraty, A. K., Kaveh, A., & Sharifi, F. (2009). Nondominated Archiving Multicolony Ant Algorithm in Time–Cost Trade-Off Optimization. *Journal of Construction Engineering and Management*, *135*(7), 668–674. doi: 10.1061/(asce)0733-9364(2009)135:7(668)

Afshar-Nadjafi, B. (2014). Resource Constrained Project Scheduling Subject to Due Dates: Preemption Permitted with Penalty. *Advances in Operations Research*, *2014*, 1–10. doi: 10.1155/2014/505716

Altintas, C., & Azizoglu, M. (2020). A Resource Constrained Project Scheduling Problem with Multi-Modes. *International Journal of Information Technology Project Management*, *11*(1), 55–70. doi: 10.4018/ijitpm.2020010104

Aminbakhsh, S., & Sonmez, R. (2016). Discrete particle swarm optimization method for the large-scale discrete time–cost trade-off problem. *Expert Systems with Applications*, *51*, 177–185. doi: 10.1016/j.eswa.2015.12.041

Azizoglu, M., Çetinkaya, F. C., & Pamir, S. K. (2015). LP relaxation-based solution algorithms for the multi-mode project scheduling with a non-renewable resource. *European J. of Industrial Engineering*, *9*(4), 450. doi: 10.1504/ejie.2015.070322

Baloi, D., & Price, A. D. (2003). Modelling global risk factors affecting construction cost performance. International Journal of Project Management, 21(4), 261-269. doi:10.1016/s0263-7863(02)00017-0

Baradaran, S., Ghomi, S. F., Ranjbar, M., & Hashemin, S. (2012). Multi-mode renewable resource-constrained allocation in PERT networks. *Applied Soft Computing*, *12*(1), 82–90. doi: 10.1016/j.asoc.2011.09.007

Birjandi, A., & Mousavi, S. M. (2019). Fuzzy resource-constrained project scheduling with multiple routes: A heuristic solution. *Automation in Construction*, *100*, 84–102. doi: 10.1016/j.autcon.2018.11.029

Brucker, P., Drexl, A., Möhring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, *112*(1), 3–41. doi: 10.1016/s0377-2217(98)00204-5

Buddhakulsomsiri, J., & Kim, D. S. (2006). Properties of multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. *European Journal of Operational Research*, *175*(1), 279–295. doi: 10.1016/j.ejor.2005.04.030

Chaleshtarti, A. S., & Shadrokh, S. (2014). A Branch and Cut Algorithm for Resource-Constrained Project Scheduling Problem Subject to Nonrenewable Resources with Pre-Scheduled Procurement. *Arabian Journal for Science and Engineering*, *39*(11), 8359–8369. doi: 10.1007/s13369-014-1319-9

Chen, P.-H., & Weng, H. (2009). A two-phase GA model for resource-constrained project scheduling. *Automation in Construction*, *18*(4), 485–498. doi: 10.1016/j.autcon.2008.11.003

Cheng, J., Fowler, J., Kempf, K., & Mason, S. (2015). Multi-mode resource-constrained project scheduling problems with non-preemptive activity splitting. *Computers & Operations Research*, *53*, 275–287. doi: 10.1016/j.cor.2014.04.018

Christodoulou, S. E., Ellinas, G., & Michaelidou-Kamenou, A. (2010). Minimum Moment Method for Resource Leveling Using Entropy Maximization. *Journal of Construction Engineering and Management*, *136*(5), 518–527. doi: 10.1061/(asce)co.1943-7862.0000149

Christofides, N., Alvarez-Valdes, R., & Tamarit, J. (1987). Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research*, *29*(3), 262–273. doi: 10.1016/0377-2217(87)90240-2

Coelho, J., & Vanhoucke, M. (2011). Multi-mode resource-constrained project scheduling using RCPSP and SAT solvers. *European Journal of Operational Research*, *213*(1), 73–82. doi: 10.1016/j.ejor.2011.03.019

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

Davis, E. W., & Heidorn, G. E. (1971). An Algorithm for Optimal Project Scheduling under Multiple Resource Constraints. *Management Science*, *17*(12). doi: 10.1287/mnsc.17.12.b803

Dayanand, N., & Padman, R. (2001). A Two Stage Search Heuristic for Scheduling Payments in Projects. *Annals of Operations Research*, 102, 197–220. doi: 10.1023/A:1010910316909

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182-197. https://doi.org/10.1109/4235.996017

Dorigo, M. (1992). Optimization, learning and natural algorithms (Unpublished doctoral dissertation). Politecnico di Milano.

Elmaghraby, S. E. (1977). *Activity networks: Project planning and control by network models*. New York: Wiley.

Elmasry, M., Zayed, T., & Hawari, A. (2019). Multi-Objective Optimization Model for Inspection Scheduling of Sewer Pipelines. Journal of Construction Engineering and Management, 145(2), 04018129. doi:10.1061/(asce)co.1943-7862.0001599

Erenguc, S. S., Ahn, T., & Conway, D. G. (2001). The resource constrained project scheduling problem with multiple crashable modes: An exact solution method. *Naval Research Logistics*, *48*(2), 107–127. doi: 10.1002/1520-6750(200103)48:2<107::aid-nav1>3.0.co;2-9

Eshtehardian, E., Afshar, A., & Abbasnia, R. (2009). Fuzzy-based MOGA approach to stochastic time–cost trade-off problem. *Automation in Construction*, *18*(5), 692–701. doi: 10.1016/j.autcon.2009.02.001

Feng, C.-W., Liu, L., & Burns, S. A. (1997). Using Genetic Algorithms to Solve Construction Time-Cost Trade-Off Problems. *Journal of Computing in Civil Engineering*, *11*(3), 184–189. doi: 10.1061/(asce)0887-3801(1997)11:3(184)

France-Mensah, J., & O'Brien, W. J. (2019). Developing a Sustainable Pavement Management Plan: Tradeoffs in Road Condition, User Costs, and Greenhouse Gas Emissions. Journal of Management in Engineering, 35(3), 04019005. doi:10.1061/(asce)me.1943-5479.0000686

Ghoddousi, P., Eshtehardian, E., Jooybanpour, S., & Javanmardi, A. (2013). Multi-mode resource-constrained discrete time–cost-resource optimization in project scheduling using non-dominated sorting genetic algorithm. *Automation in Construction*, *30*, 216–227. doi: 10.1016/j.autcon.2012.11.014

Goldberg. (1989). Genetic algorithms in search, optimization, and machine learning. *Choice Reviews Online*, *27*(02), 27-0936-27-0936. https://doi.org/10.5860/choice.27-0936

Hariga, M., & El-Sayegh, S. M. (2011). Cost Optimization Model for the Multiresource Leveling Problem with Allowed Activity Splitting. *Journal of Construction Engineering and Management*, *137*(1), 56–64. doi: 10.1061/(asce)co.1943-7862.0000251

Hariga, M., Shamayleh, A., & El-Wehedi, F. (2016). Integrated time-cost tradeoff and resources leveling problems with allowed activity splitting. International Transactions in Operational Research, 26(1), 80-99. doi:10.1111/itor.12329

Harris, R. B. (1978). Precedence and arrow networking techniques for construction. New York: John Wiley & Sons.

Harris, R. B. (1990). Packing Method for Resource Leveling (Pack). *Journal of Construction Engineering and Management*, *116*(2), 331–350. doi: 10.1061/(asce)0733-9364(1990)116:2(331)

Hartmann, S., & Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, *207*(1), 1–14. doi: 10.1016/j.ejor.2009.11.005

Hegazy, T. (1999). Optimization of Resource Allocation and Leveling Using Genetic Algorithms. *Journal of Construction Engineering and Management*, *125*(3), 167–175. doi: 10.1061/(asce)0733-9364(1999)125:3(167)

Hegazy, T., & Ersahin, T. (2001). Simplified Spreadsheet Solutions. II: Overall Schedule Optimization. *Journal of Construction Engineering and Management*, *127*(6), 469–475. doi: 10.1061/(asce)0733-9364(2001)127:6(469)

Hegazy, T., & Menesi, W. (2012). Heuristic Method for Satisfying Both Deadlines and Resource Constraints. *Journal of Construction Engineering and Management*, *138*(6), 688–696. doi: 10.1061/(asce)co.1943-7862.0000483

Heilmann, R. (2001). Resource–constrained project scheduling: a heuristic for the multi–mode case. *OR-Spektrum*, *23*(3), 335–357. doi: 10.1007/pl00013354

Heilmann, R. (2003). A branch-and-bound procedure for the multi-mode resource-constrained project scheduling problem with minimum and maximum time lags. *European Journal of Operational Research*, *144*(2), 348–365. doi: 10.1016/s0377-2217(02)00136-4

Herroelen, W., & Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, *165*(2), 289–306. doi: 10.1016/j.ejor.2004.04.002

Hiyassat, M. A. S. (2001). Applying Modified Minimum Moment Method to Multiple Resource Leveling. *Journal of Construction Engineering and Management*, *127*(3), 192–198. doi: 10.1061/(asce)0733-9364(2001)127:3(192)

Hussein, B., & Moselhi, O. (2019). An evolutionary stochastic discrete time-cost trade-off method. *Canadian Journal of Civil Engineering*, *46*(7), 581–600. doi: 10.1139/cjce-2018-0352

Kadam, S. U., & Kadam, N. S. (2014). Solving resource-constrained project scheduling problem by genetic algorithm. *2014 2nd International Conference on Business and Information Management (ICBIM)*. doi: 10.1109/icbim.2014.6970966

Kaiafa, S., & Chassiakos, A. P. (2015). A Genetic Algorithm for Optimal Resource-driven Project Scheduling. *Procedia Engineering*, *123*, 260–267. doi: 10.1016/j.proeng.2015.10.087

Kalhor, E., Khanzadi, M., Eshtehardian, E., & Afshar, A. (2011). Stochastic time–cost optimization using non-dominated archiving ant colony approach. *Automation in Construction*, *20*(8), 1193–1203. doi: 10.1016/j.autcon.2011.05.003

Karaa, F. A., & Nasr, A. Y. (1986). Resource Management in Construction. *Journal of Construction Engineering and Management*, *112*(3), 346–357. doi: 10.1061/(asce)0733-9364(1986)112:3(346)

Khan, A., H(2017) NSGA-II.py (Version 1.0) [Python]. https://github.com/haris989/NSGA-II.git

Klir, G. J., Wang, Z., & Harmanec, D. (1997). Constructing fuzzy measures in expert systems. *Fuzzy Sets and Systems*, *92*(2), 251-264. https://doi.org/10.1016/s0165-0114(97)00174-7

Lee, J., Madanat, S., & Reger, D. (2016). Pavement systems reconstruction and resurfacing policies for minimization of life-cycle costs under greenhouse gas emissions constraints. Transportation Research Part B: Methodological, 93, 618-630. doi:10.1016/j.trb.2016.08.016

Leu, S.-S., & Yang, C.-H. (1999). GA-Based Multicriteria Optimal Model for Construction Scheduling. *Journal of Construction Engineering and Management*, *125*(6), 420–427. doi: 10.1061/(asce)0733-9364(1999)125:6(420)

Li, F., Lai, C., & Shou, Y. (2011). Particle swarm optimization for preemptive project scheduling with resource constraints. *2011 IEEE International Conference on Industrial Engineering and Engineering Management*. doi: 10.1109/ieem.2011.6118040

Li, H., & Zhang, H. (2013). Ant colony optimization-based multi-mode scheduling under renewable and nonrenewable resource constraints. *Automation in Construction*, *35*, 431–438. doi: 10.1016/j.autcon.2013.05.030

Liang, Y., Cui, N., Hu, X., & Demeulemeester, E. (2019). The integration of resource allocation and time buffering for bi-objective robust project scheduling. *International Journal of Production Research*, 1–16. doi: 10.1080/00207543.2019.1636319

Martin, O., & Klir, G. J. (2007). Defuzzification as a special way of dealing with retranslation. International Journal of General Systems, 36(6), 683-701. doi:10.1080/03081070701456088

Memon, A., Rahman, I., & Abdullah, M. (2010). Factors Affecting Construction Cost in Mara Large Construction Project. Factors Affecting Construction Cost in Mara Large Construction Project.

Menesi, W., & Hegazy, T. (2015). Multimode Resource-Constrained Scheduling and Leveling for Practical-Size Projects. *Journal of Management in Engineering*, *31*(6), 04014092. doi: 10.1061/(asce)me.1943-5479.0000338

Moselhi, O., & Alshibani, A. (2013). Schedule compression using fuzzy set theory and contractor judgment. *Journal of Information Technology in Construction*, (18)

Moselhi, O., & Roofigari-Esfahan, N. (2013). Project schedule compression: a multi-objective methodology. *Construction Innovation*, *13*(4), 374–393. doi: 10.1108/ci-03-2011-0010

Moukrim, A., Quilliot, A., & Toussaint, H. (2015). An effective branch-and-price algorithm for the Preemptive Resource Constrained Project Scheduling Problem based on minimal Interval Order Enumeration. *European Journal of Operational Research*, *244*(2), 360–368. doi: 10.1016/j.ejor.2014.12.037

Ouma, Y. O., Yabann, C., Kirichu, M., & Tateishi, R. (2014). Optimization of Urban Highway Bypass Horizontal Alignment: A Methodological Overview of Intelligent Spatial MCDA Approach Using Fuzzy AHP and GIS. Advances in Civil Engineering, 2014, 1-26. doi:10.1155/2014/182568

Peteghem, V. 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. doi: 10.1016/j.ejor.2009.03.034

Roca, J., Pugnaghi, E., & Libert, G. (2008). Solving an Extended Resource Leveling Problem With Multi-Objective Evolutionary Algorithms. *Computers & Industrial Engineering*

Salama, T., & Moselhi, O. (2019). Multi-objective optimization for repetitive scheduling under uncertainty. *Engineering, Construction and Architectural Management*, *26*(7), 1294–1320. doi: 10.1108/ecam-05-2018-0217

Singh, A. (2014). Resource Constrained Multi-project Scheduling with Priority Rules & Analytic Hierarchy Process. *Procedia Engineering*, *69*, 725–734. doi: 10.1016/j.proeng.2014.03.048

Slowinski, R. (1980). Two Approaches to Problems of Resource Allocation among Project Activities -- A Comparative Study. *The Journal of the Operational Research Society*, *31*(8), 711. doi: 10.2307/2581688

Son, J., & Mattila, K. G. (2004). Binary Resource Leveling Model: Activity Splitting Allowed. Journal of Construction Engineering and Management, 130(6), 887-894. doi:10.1061/(asce)0733-9364(2004)130:6(887)

Sprecher, A., & Drexl, A. (1998). Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. *European Journal of Operational Research*, *107*(2), 431–450. doi: 10.1016/s0377-2217(97)00348-2

Srinivas, N., & Deb, K. (1994). Muiltiobjective optimization using Nondominated sorting in genetic algorithms. *Evolutionary Computation*, *2*(3), 221-248. https://doi.org/10.1162/evco.1994.2.3.221

Tao, R., & Tam, C.-M. (2013). System reliability theory based multiple-objective optimization model for construction projects. *Automation in Construction*, *31*, 54–64. doi: 10.1016/j.autcon.2012.11.040

Tawalare, A., & Lalwani, R. (2012). Resource Leveling in Construction Projects using Re- Modified Minimum Moment Approach. *World Academy of Science, Engineering and Technology*, 6(2), 200–202.

Tirkolaee, E. B., Goli, A., Hematian, M., Sangaiah, A. K., & Han, T. (2019). Multi-objective multi-mode resource constrained project scheduling problem using Pareto-based algorithms. *Computing*, *101*(6), 547–570. doi: 10.1007/s00607-018-00693-1

Toğan, V., & Eirgash, M. A. (2018). Time-Cost Trade-off Optimization of Construction Projects using Teaching Learning Based Optimization. *KSCE Journal of Civil Engineering*, *23*(1), 10–20. doi: 10.1007/s12205-018-1670-6

Torres-Machi, C., Osorio-Lird, A., Chamorro, A., Videla, C., Tighe, S. L., & Mourgues, C. (2018). Impact of environmental assessment and budgetary restrictions in pavement maintenance decisions: Application to an urban network. Transportation Research Part D: Transport and Environment, 59, 192-204. doi:10.1016/j.trd.2017.12.017

Węglarz, J., Józefowska, J., Mika, M., & Waligóra, G. (2011). Project scheduling with finite or infinite number of activity processing modes – A survey. *European Journal of Operational Research*, *208*(3), 177–205. doi: 10.1016/j.ejor.2010.03.037

Xu, X., Hao, J., & Zheng, Y. (2020). Multi-Objective Artificial Bee Colony Algorithm for Multi-Stage Resource Leveling Problem in Sharing Logistics Network. *Computers & Industrial Engineering*, 142, doi: 10.1016/j.cie.2020.106338

Yandamuri, S. R., Srinivasan, K., & Murty Bhallamudi, S. (2006). Multiobjective Optimal Waste Load Allocation Models for Rivers Using Nondominated Sorting Genetic Algorithm-II. Journal of

Water Resources Planning and Management, 132(3), 133–143. doi:10.1061/(asce)0733-9496(2006)132:3(133)

Zadeh, L. (1965). Fuzzy sets. *Information and Control*, *8*(3), 338-353.https://doi.org/10.1016/s0019-9958(65)90241-x

Zahraie, B., & Tavakolan, M. (2009). Stochastic Time-Cost-Resource Utilization Optimization Using Nondominated Sorting Genetic Algorithm and Discrete Fuzzy Sets. *Journal of Construction Engineering and Management*, *135*(11), 1162–1171. doi: 10.1061/(asce)co.1943-7862.0000092

Zheng, D. X. M., & Ng, S. T. (2005). Stochastic Time–Cost Optimization Model Incorporating Fuzzy Sets Theory and Nonreplaceable Front. *Journal of Construction Engineering and Management*, *131*(2), 176–186. doi: 10.1061/(asce)0733-9364(2005)131:2(176)

Zheng, D. X. M., Ng, S. T., & Kumaraswamy, M. M. (2004). Applying a Genetic Algorithm-Based Multiobjective Approach for Time-Cost Optimization. *Journal of Construction Engineering and Management*, *130*(2), 168–176. doi: 10.1061/(asce)0733-9364(2004)130:2(168)

Zheng, D. X. M., Ng, S. T., & Kumaraswamy, M. M. (2005). Applying Pareto Ranking and Niche Formation to Genetic Algorithm-Based Multiobjective Time–Cost Optimization. *Journal of Construction Engineering and Management*, *131*(1), 81–91. doi: 10.1061/(asce)0733-9364(2005)131:1(81)

Appendix I:

```
# Program Name: NSGA-II.py
# Description: This is a python implementation of Prof. Kalyanmoy Deb's popular N
SGA-II algorithm
# Author: Haris Ali Khan
# Supervisor: Prof. Manoj Kumar Tiwari

#Function to find index of list
def index_of(a,list):
    for i in range(0,len(list)):
        if list[i] == a:
            return i
    return -1

#Function to sort by values
def sort_by_values(list1, values):
    sorted_list = []
    while(len(sorted_list)!=len(list1)):
        if index_of(min(values),values) in list1:
            sorted_list.append(index_of(min(values),values))
        values[index_of(min(values),values)] = math.inf
    return sorted_list

#Function to carry out NSGA-II's fast non dominated sort
def fast_non_dominated_sort(values1, values2):
    S=[[] for i in range(0,len(values1))]
    front = [[]]
    n=[0 for i in range(0,len(values1))]
    rank = [0 for i in range(0, len(values1))]

    for p in range(0,len(values1)):
        S[p]=[]
        n[p]=0
        for q in range(0, len(values1)):
            if (values1[p] > values1[q] and values2[p] > values2[q]) or (values1[
p] >= values1[q] and values2[p] > values2[q]) or (values1[p] > values1[q] and val
ues2[p] >= values2[q]):
                if q not in S[p]:
                    S[p].append(q)
            elif (values1[q] > values1[p] and values2[q] > values2[p]) or (values
1[q] >= values1[p] and values2[q] > values2[p]) or (values1[q] > values1[p] and v
alues2[q] >= values2[p]):
                n[p] = n[p] + 1
```

```
        if n[p]==0:
            rank[p] = 0
            if p not in front[0]:
                front[0].append(p)

    i = 0
    while(front[i] != []):
        Q=[]
        for p in front[i]:
            for q in S[p]:
                n[q] =n[q] - 1
                if( n[q]==0):
                    rank[q]=i+1
                    if q not in Q:
                        Q.append(q)
        i = i+1
        front.append(Q)

    del front[len(front)-1]
    return front

#Function to calculate crowding distance
def crowding_distance(values1, values2, front):
    distance = [0 for i in range(0,len(front))]
    sorted1 = sort_by_values(front, values1[:])
    sorted2 = sort_by_values(front, values2[:])
    distance[0] = 4444444444444444
    distance[len(front) - 1] = 4444444444444444
    for k in range(1,len(front)-1):
        distance[k] = distance[k]+ (values1[sorted1[k+1]] - values2[sorted1[k-
1]])/(max(values1)-min(values1)+1000000)
    for k in range(1,len(front)-1):
        distance[k] = distance[k]+ (values1[sorted2[k+1]] - values2[sorted2[k-
1]])/(max(values2)-min(values2)+1000000)
    return distance
```

```
"""
#!/usr/bin/env python

2013.3.12 CKS

A simple critical path method implementation.

http://en.wikipedia.org/wiki/Critical_path_method

To run a unittest:

    python criticalpath.py Test.test_model
"""

    def get_critical_path(self, as_item=False):
        """
        Finds the longest path in among the child nodes.
        """
        if self._critical_path is not None:
            # Returned cached path.
            return self._critical_path[1]
        longest = None
        q = [(_.duration, [_], set([_])) for _ in self.first_nodes]
        while q:
```

```
            item = length, path, priors = q.pop(0)
            if longest is None:
                longest = item
            else:
                try:
                    longest = max(longest, item)
                except TypeError:
                    longest = longest
            for to_node in path[-1].to_nodes:
                if to_node in priors:
                    continue
                q.append((length+to_node.duration, path+[to_node], priors.union([
to_node])))
        if longest is None:
            return
        elif as_item:
            return longest
        else:
            return longest[1]

    def print_times(self):
        w = 7
        print("""
```

```
+{border}+

|{blank} DUR={dur} {blank}|

+{border}+

|ES={es}|{blank}|EF={ef}|

|{segment}|{name}|{segment}|

|LS={ls}|{blank}|LF={lf}|

+{border}+

|{blank}DRAG={drag}{blank}|

+{border}+

""".format(

            blank=' '*w,

            segment='-'*w,

            border='-'*(w*3 + 2),

            dur=str(self.duration).ljust(w-4),

            es=str(self.es).ljust(w-3),

            ef=str(self.ef).ljust(w-3),

            name=str(self.name).center(w),

            ls=str(self.ls).ljust(w-3),

            lf=str(self.lf).ljust(w-3),

            drag=str(self.drag).ljust(w-5),

        ))
```

```
  # Author: Moaaz Elkabalawy
  # Description: Critical path method for generating project Gantt Chart and
                 performing non-critical activities splitting

l1 =[] # early start finishes for activities
l2 = []# l2 is early finishes for activities
l3 = []# l3 is late finishes for activities

for nod in node_list:
    l1.append(nod.es)

    l2.append(nod.ef)

    l3.append(nod.lf)
n = len(node_list) # number of activities
m = int(p.duration) #p.duration # duration of the project

from numpy import zeros
A = zeros([n,m])

i = 0
j = 0
r = 0
z = 0

while (i<=n): # this function is used to generate the project Gantt chart
    A[i][j]=1
    j = j+1
    if (j>=l2[r]):
        r = r + 1
        z = z + 1
        j = l1[z]
        i = i + 1
        if (z==n):
            break

B = A.copy()

i = 0
r = 0
j = 0
z = 0
```

```
    while (i<=n):  # this function is used to assign the needed resources for
each activity on each period of the project
        while (A[i][j]==1) and (j<=p.duration-1):
            A[i][j]= ind_res[r]
            j= j + 1
            if j == p.duration:
                break
        z = z + 1
        j = l1[z]
        r= r + 1
        i= i + 1
        if (z==n):
            break


    i = 0
    RD = [sum([row[i] for row in A]) for i in range(0,len(A[0]))] # resource
demand in each period of the project
    RA = np.zeros((p.duration))
    RR = np.zeros((p.duration))

    t = 1
    i = 1
    while(t<p.duration-1): # Aquiring resources in each period of the project
        RA[t]= max (0,RD[i]-RD[i-1])
        i= i + 1
        t= t + 1
    RA[0]= RD[1]

    t = 1
    i = 0
    while(t<p.duration-1): # Releasing resources in each period of the project
        RR[t]= max (0,RD[i]-RD[i+1])
        i= i + 1
        t= t + 1
    RR[0]= 0

i = len(p.get_critical_path())
    for i1, act in enumerate(A): # splitting function for non-
critical activities)
        for j1, day in enumerate(act):
            #temp as activity counter or row counter
            temp = i1
            while RD[j1] > RS and temp >= i and l2[temp] < l3[temp] :
```

```
                #put the resource in the early finish
                A[temp][l2[temp]] = A[temp][j1]

                #put the day resource for the non-critical activity zero
                A[temp][j1] = 0

                #the early finish is increased by one
                l2[temp] = l2[temp]+1
                RD = [sum([row[k] for row in A]) for k in range(0,len(A[0]))]
                temp = temp + 1

    Acopy = A.copy()*0
    for i, act in enumerate(A):
        for j, day in enumerate(act):
            if day > 0:
                Acopy[i][j]= 1

    split_count = 0
    for i, act in enumerate(Acopy):
        for j, day in enumerate(act):
            if j>0:
                if day < act[j-1] :
                    split_count = split_count + abs(day-act[j-1])
                if j == p.duration-1 and day == 1:
                    split_count = split_count + 1
        split_count = split_count - 1
```

```
      # Author: Moaaz Elkabalawy


      # Description: Triangular Fuzzy numbers representation
                     with the alpha cut method

ind_dur = []


    j = 0
    result = []

    for k in Y:
        ind_dur.append(duration[j][k])
        j = j + 1
    j = 0
    while j < n:
        k = 0
        x = np.arange(ind_dur[j][k], ind_dur[j][k+2]+1)
        mfx = skfuzzy.trimf(x,(ind_dur[j]))
        xxx = fuzz.lambda_cut_boundaries(x,mfx, Choosen_risk_level)
        result.append(xxx)
        j = j + 1

    i = 0
    temp_postive = []
    temp_negative= []
    while i < n:
        if Choosen_risk_level != 1:
            temp_postive.append(result[i][0])
            temp_negative.append(result[i][1])
        if Choosen_risk_level == 1:
            temp_postive.append(result[i][0])
            temp_negative.append(result[i][0])
        i = i + 1
    alpha_final= []
    i = 0
    while i <n:
        temp = (temp_postive[i] + temp_negative[i])/2
        alpha_final.append(temp)
        i = i + 1

    set_of_durations = []
    p = Node('project')
```

```
    node_list= []
    i = 0
    while i< int(number_of_act):

        node_list.append(p.add(Node(act_list[i], duration=int(alpha_final[i]) ) ) )
)


        i = i +1

    i = 0
    while i< Number_of_prec_relations:

        p.link(p.lookup_node(act1[i]), p.lookup_node(act2[i]))

        i = i + 1
    p.update_all()
    set_of_durations.append(p.duration)
```