Decision Support NSGA-II Optimization Method for Resource-Constrained Schedule Compression with Allowed Activity Splitting

Decision Support NSGA-II Optimization Method for Resource-Constrained Schedule Compression with Allowed Activity Splitting

M Elkabalawy[1] **and A Al-Sakkaf**[1, 2,*]

[1] Department of Building Engineering, Concordia University, Montreal, QC, Canada.
[2] Department of Architecture & Environmental Planning, College of Engineering & Petroleum, Hadhramout University, Mukalla, Yemen.

[*]Corresponding author: abobakr.alsakkaf@concordia.ca

Abstract. In the course of a construction project, the project manager's task is to ensure timely and cost-effective execution of the job. However, it is common that delays and over-budgeting to be experienced during the project execution. This schedule acceleration requires resource planning to account for the project's limited resources. Therefore, this study proposes an integrated method that allows for joint consideration of project scheduling and resource planning while accounting for activity splitting. The objective is to determine the project's optimal cost and duration while considering some input parameters such as the crew's size and project's activities' cost and duration. The proposed method utilized the Genetic Algorithm (GA) to optimize the project duration and cost. Accordingly, the Weighted Sum was used as a multi-criteria decision support method to choose an optimal solution from the optimization results. The developed scheduling and optimization method is coded in Python as a stand-alone, automated, computerized tool to facilitate its application. A numerical example, utilizing the developed method, is employed to show the method's robustness and assess its performance against other previously developed methods. Results indicated the developed method's dominance in finding optimal solutions in a reasonable time avoiding local minima entrapment.

1. Background

One technique that has been explored both in academia and industry to overcome construction delays is schedule compression. Unwanted arguments between contractors and clients stem most likely from construction delays. Many of the World Bank-funded construction projects (1120 out of 1778) surpass their proposed budgets [1]. The actual time and cost for many construction projects in developing nations often surpass the projected plan for both parameters [2]. The time-cost optimization problem (TCO) is typically referred to as accelerating the project's completion time. TCO models' objective is to reduce the time for a project and maintain its allotted budget [3]. The relation between time and cost is linear. The use of additional resources leads to lesser work time and higher costs. Finding the duration with minimal cost to be incurred has always been a topic of interest (Figure 1). Existing literature has proposed various methodologies and algorithms, including objective functions similar to the project's defined objectives, to tackle the schedule optimization problem. The proposed algorithms include exact methods (linear/integer or dynamic programming), heuristic algorithms, and meta-heuristic or evolutionary algorithms. Objective functions can be classified as time-based, cost-based, and quality-based objective functions. Metaheuristics and evolutionary algorithms have been extensively studied due to the increased size of the TCO problem, project activities and execution modes. Metaheuristics

algorithms, in the face of incomplete information or reduced computation capacity, are tasked to seek out, generate or choose a heuristic to solve the optimization problem.

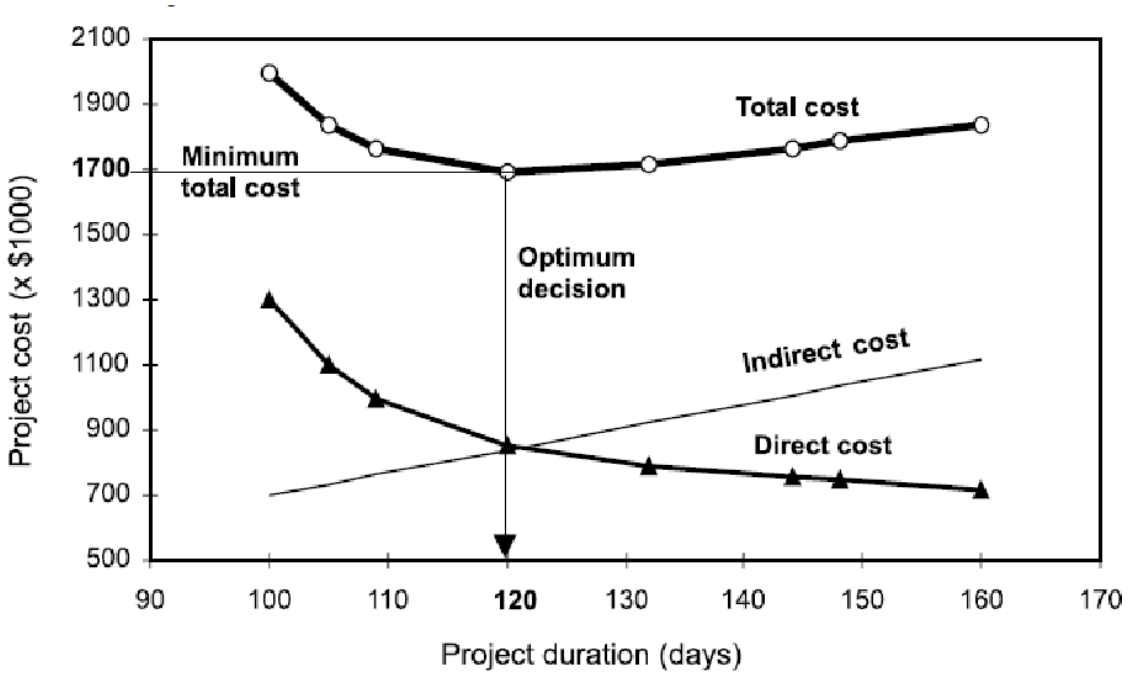

Figure 1. Time and cost trade-off [4].

Examples of the metaheuristics method include Genetic Algorithm (GA), Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO). Aminbakhsh and Sonmez [5] developed a particle swarm optimization for medium and large-scaled TCTPs. Their efficiently developed methodology proved to outperform the previously proposed methods in solution quality and computation time, especially for large-scale projects. Toğan and Eirgash [6] 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 effectively generates optimal or suboptimal solutions for TCTP in the construction engineering and management field than the previously proposed metaheuristic methods. Another difficulty that contractors face during schedule compression is accounting for the limited resources, otherwise known as resource-constrained project scheduling. To reduce cost and project length, several methods have been explored both in academia and industry. In the late 1950s, Morgan Walker and James Kelley developed the critical path method (CPM) to improve cost-related aspects of scheduling. This technique has been employed for project scheduling in many construction projects. Although CPM accounts for time usage and identifies ways to reduce the project's length, it does not consider the limited resources available for the project. Accordingly, resource-constrained scheduling models were developed to overcome this limitation. The aim of scheduling models that involve limited resources is to develop optimal schedules that depend on available resources and minimal or approved length of time [7]. The assumption in the primary form of the resource-constrained project scheduling problem is that every activity can be performed by one method within a determined processing time with one renewable resource. The more elaborate form of RCPSP is the multi-mode resource-constrained project scheduling problem (MMRCPSP) in which several techniques (alternatives) are available to execute project activities. Each execution mode has its unique duration and resource needs [8]. MMRCPSP was first introduced by Elmaghraby [9] where the author assumed that an activity j must be performed and analyzed by a mode $m \in Mj$ until completion. Resources categories were first defined by Slowinski [10] as renewable and non-renewable resources based on their availability. Renewable resources represent labor, machinery, equipment, and non-

renewable resources represented as money. A resource is renewable if its quantity stays the same throughout the entire period of the project, while a non-renewable resource is one whose availability during the project period is limited [11,12]. Li and Zhang [13] considered the renewable and non-renewable resources in an ant colony optimization algorithm to solve the resource-constrained scheduling problem. Azizoglu, Çetinkaya, and Pamir [14] developed a linear programming relaxation-based heuristic solution algorithm involving a nonrenewable resource. The algorithm was implemented in a project scheduling problem. Altintas and Azizoglu [15] discussed the multi-mode resource-constrained and discrete time-cost trade-off problem. They optimized activity execution modes progressively using a non-renewable resource. Chaleshtarti and Shadrokh [16] studied both renewable and non-renewable resources as an extended form of the resource-constrained scheduling problem. They proposed a branch and cut algorithm with some techniques that shorten the size of the models 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. The basic assumption in the traditional resource-constrained project scheduling problem (RCPSP) is that activities cannot be interrupted once started. Therefore, 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. In reality, interruptions could occur 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. Activity splitting is employed for project improvement in the face of limited or unavailable resources [17,18]. Peteghem and Vanhoucke [19] solved the pre-emption 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 and introduces a mode improvement procedure. Preemption without a penalty was discussed by Moukrim, Quilliot and Toussaint [20]. 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 [21]. 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 in time with a constant setup penalty and without setup time. On the contrary, Li, Lai and Shou [22] 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 four designed types of particle representations. Cheng, Fowler, Kempf and Mason [23] introduced the non-preemptive activity splitting to deal with the varying capacity of renewable resources. 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. Table 1 presents a critique of relevant literature, and it is evident that existing scheduling methods:

1. Doesn’t consider the project limited resources.
2. Doesn’t account for activity splitting.
3. Doesn’t consider the impact of delay penalties.

To address the above needs, a schedule compression method that considers the availability of limited resources is proposed for activity splitting, project scheduling and resource planning. The method aims to reduce the project duration and cost, including direct and indirect cost, delay penalty, and activity splitting cost.

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

Criteria	**Reference**						
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Methods	MILP	ACO	GA	PSO	SA	MAWA-TLBO	MILP
Time-Cost Optimization				✔		✔	
Consider Direct Project Cost				✔	✔	✔	
Resource-Constrained Scheduling	✔	✔	✔		✔		✔
Consider Delay Penalty/ Bonus					✔		
Allow Activity Splitting			✔		✔		

NOTES: (TLBO) Teaching-Learning Based Optimization, (ACO) Ant Colony Optimization, (SA) Simulated Annealing Algorithm, (PSO) Particle Swarm Optimization, (MAWA-TLBO) Modified Adaptive Weight Approach and Teaching Learning Based Optimization, (MILP) Mixed Integer Linear Programming; (1) Azizoglu, Çetinkaya and Pamir [14], (2) Li and Zhang [13] , (3) Peteghem and Vanhoucke [19], (4) Aminbakhsh and Sonmez [5], (5) Afshar-Nadjafi [21], (6) Toğan and Eirgash [6], (7) Altintas and Azizoglu [15].

2. Methodology

This study provides a solution to resolving the resource-constrained project scheduling problem that permits the user to select an optimal solution from the resulted set of non-dominated solutions. The developed method design relies on the integration of four modules: scheduling module using a modified critical path method that considers the project resources; cost calculation module that takes into account all costs, direct and indirect, delay penalty, and splitting cost; multi-objective optimization module using the elitist non-dominated sorting genetic algorithm (NSGA-II); and decision-support module. The developed method is programmed in Python as a computer application to facilitate project activities' rescheduling (in an iterative way) and project schedule optimization. The steps employed to generate an optimal schedule and project cost using the developed modules are summarized in the flow chart, shown in Figure 2.

2.1. Scheduling Module

For this module, the user identifies the activities execution modes that define the normal and crashed durations, related resource requirements, and precedence relationships. The developed scheduling module uses a Python critical path library, Criticalpath, to define the critical and non-critical activities along with their early and late start, and early and late finish. Accordingly, the project duration (T) is calculated. The critical path library was then modified to sustain the resource-constrained scheduling environment by assigning the required resources to the project activities. Accordingly, it generates a Gantt chart for the project that calculates the resource demand for each project period. The developed

module would reschedule the non-critical activities within their floats and perform non-critical activity splitting when necessary based on the resource demand calculations and resource availability.

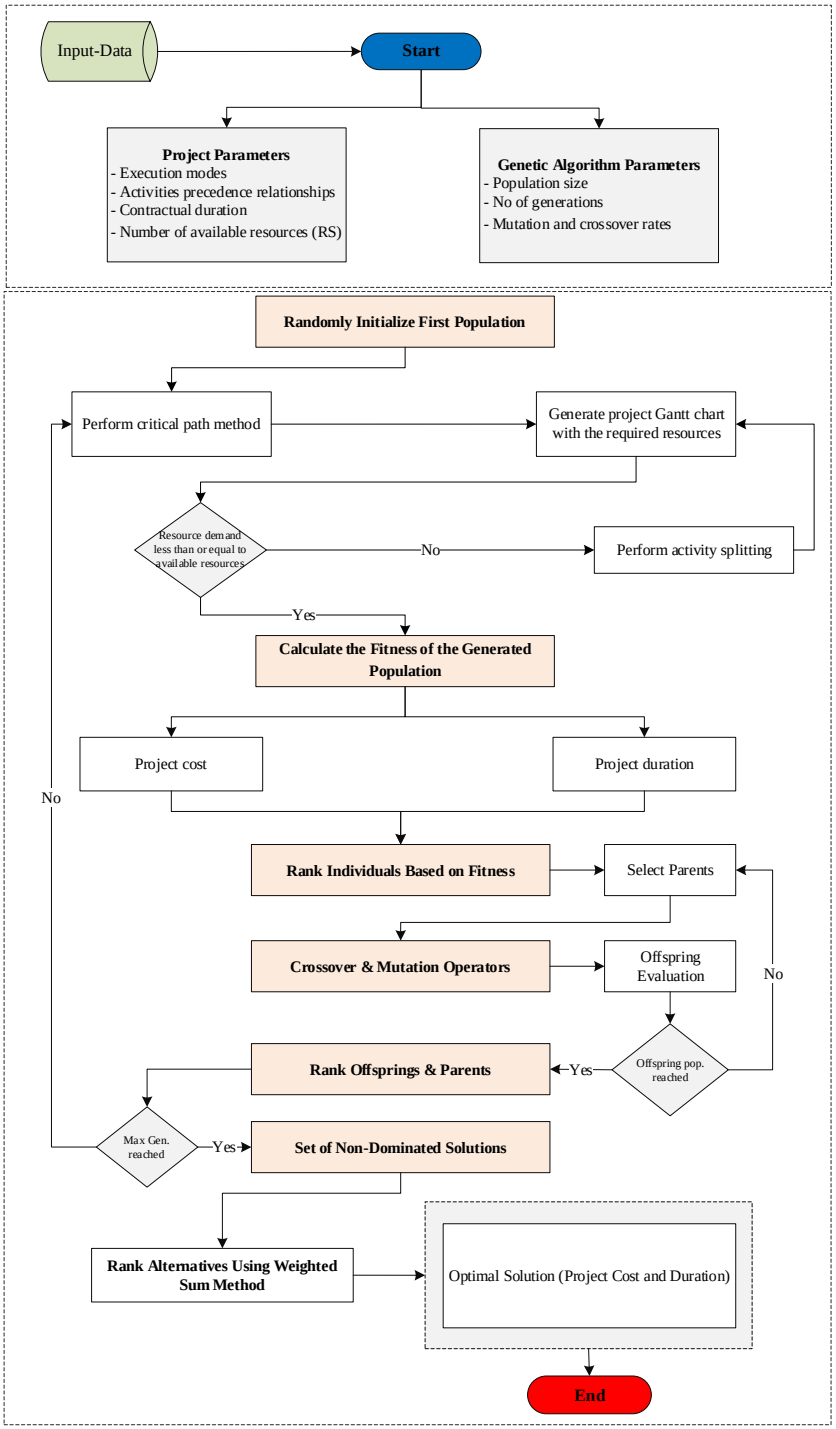

Figure 2. Flow chart of the proposed method.

To demonstrate the scheduling module's splitting function, we will use a similar example, as in Son and Mattila [24], when multiple resources are available. A single machine was employed in all activities. The input data and project network are shown in Table 2 and Figure 3, 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

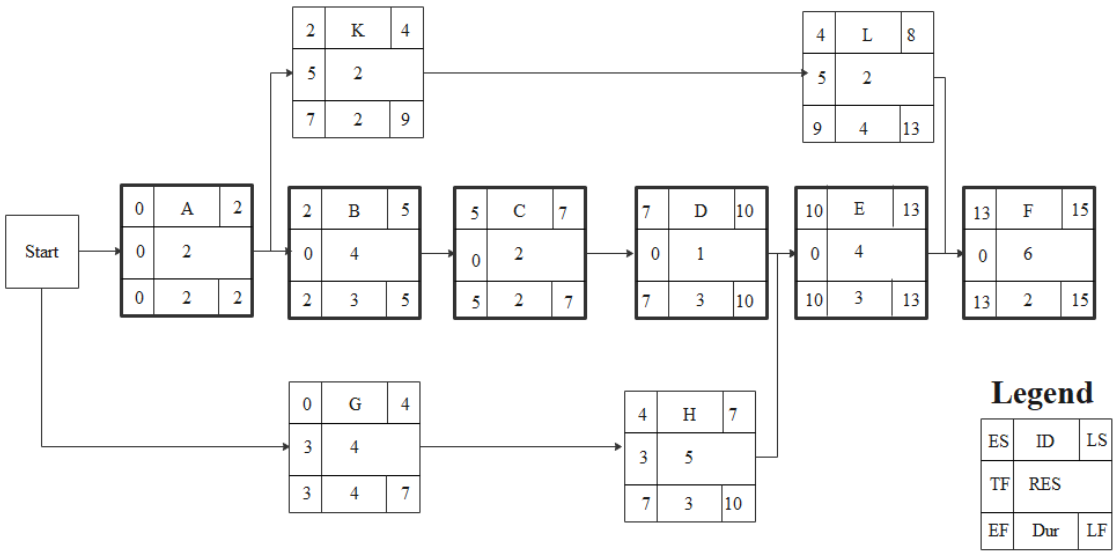

Figure 3. Example project network.

As seen in Figure 3, 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. The Gantt chart for the previously mentioned example before splitting is shown in Figure 4. The project schedule has nine available resources for a 15-day period where the resource demand was higher than the available resources in periods 3, 4, and 5. The ones and zeros 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 is active from periods 1 to 4. However, it can be split within its floats based on 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 4. The example initial Gantt chart.

A Gantt chart for a feasible project schedule with splitting is generated and shown in Figure 5. 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 the variable Y_{tj} was equal to zero in these periods (binary variable equals to one when activity j is progressing at time t; t= $ESj\ to\ LFj$ and zero otherwise). 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 resource availability in periods 4 and 5. In this case, splitting costs would not be imposed since the activity has not yet started.

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 5. The example final Gantt chart.

2.2. Cost Calculation Module

For this module, the project cost (PC) is calculated using the direct cost, indirect cost, splitting cost, and delay penalty/opportunity cost. The direct cost is the summation of the input cost of the project's activities. The indirect cost is the input indirect cost rate multiplied by the duration of the project. The splitting cost is the summation of the occurred splits multiplied by the input split cost, representing the extra costs associated with shutting down the activity and later restarting it. Finally, the delay penalty/Opportunity cost is the difference between the input contractual duration and the actual schedule duration multiplied by the input bonus payment or the delay penalty, representing the advantage of finishing the project before or after the planned date.

2.3. NSGA-II Multiobjective Optimization Module

Optimization, an area in operations research, is applied in problem-solving and enables better decision-making. Optimization utilizes certain constraints to minimize or maximize a set objective and generate an optimum solution [25]. The developed optimization method is formulated as a multiobjective optimization problem and searches for non-dominated resource-constrained schedules that minimize the total duration and total cost. The objective functions can be represented by Equations 1 and 2.

$$\text{Minimize } T = \sum_{j=1}^{NAC} \sum_{m=1}^{NM_j} Y_{jm} D_{jm}; \quad (1)$$

$$\text{Minimize } PC = \sum_{j=1}^{NA} [SC_j NS_j] + \sum_{j=1}^{NA} \sum_{m=1}^{NM_j} [Y_{jm} DC_{jm}] + IC * T + [(T - TC)B] \text{ where;} \quad (2)$$

- NA = number of activities
- NAC = number of critical activities
- NM_j = number of modes for executing activity j; j=1,2,…….N under mode m and zero otherwise
- D_{jm} = duration of activity j running in mode m; m=1,2…. Mj time t; t= $ESj\ to\ LFj$
- T = actual project duration

- B = delay penalty/ Opportunity cost
- PC = project cost
- TC = contract project duration
- SC_j = Splitting cost of activity j
- NS_j = number of times activity j is split.
- IC = indirect cost
- DC_{jm} = direct cost of activity j under mode m
- Y_{jm} = binary variable equals to 1 when activity j is performed under mode m and zero otherwise

The developed 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.

The main reason for using a multiobjective evolutionary algorithm (MOEA) optimization is to generate a Pareto optimal set of solutions at each simulation run. Furthermore, NSGA-II was elected by researchers over other MOEAs because of its credibility and tested performance in several comparative studies [26]. Deb, Pratap, Agarwal, and Meyarivan [27] proposed NSGA-II to improve the NSGA complex algorithm, suggested by Goldberg in 1989, by providing a fast, more efficient elitist multiobjective algorithm. A genetic algorithm works by encoding parent solutions into chromosomes to generate the initial generation. 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 multiobjective 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. The traditional genetic algorithm utilizes the selection, crossover, and mutation operators to generate a new population. The developed code for the NSGA-II method utilized some extracted NSGA-II functions (i.e., fast non-dominated sorting function, sorting solution, and crowding distance calculation function) from a program code written by Khan [28]. The genetic algorithm population structure for the developed optimization method is shown in Figure 6.

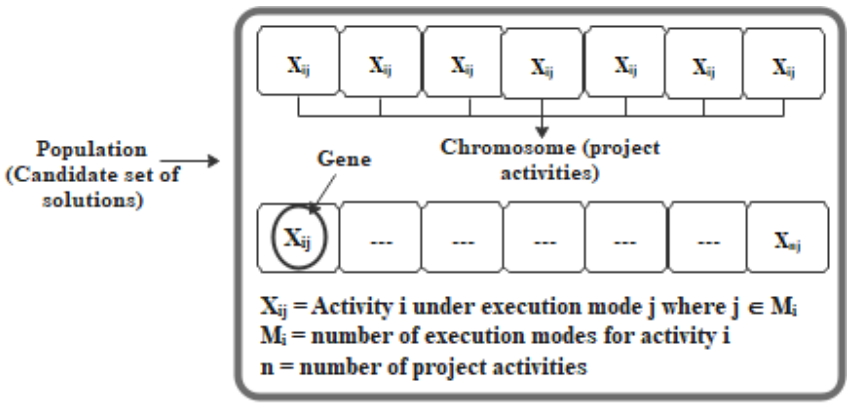

Figure 6. Description of the Genetic Algorithm population.

2.4. Decision-Support Module

Multi-criteria decision-making (MCDM) methods are very important techniques that permit considering different attributes for ranking the different scenarios and present a systematic approach to help decision-makers choose the best scenario among the generated ones [29]. The developed decision-support module utilizes the weighted sum method as an MCDM method. The developed module would help project managers select the most feasible project schedule from the generated NSGA-II Pareto front for time and cost solutions. The weighted sum method is based on calculating a preference index for each alternative. The highest preference index during maximization (and the lowest during minimization) is considered the best alternative. The preference index of each alternative can be computed using Equation 3.

$$\mathbf{P_i} = \sum_{\mathbf{j=1}}^{\mathbf{n}} \mathbf{f_{ij}} * \mathbf{w_j}(\mathbf{1} \le \mathbf{i} \le \mathbf{m}, \mathbf{1} \le \mathbf{j} \le \mathbf{n}), \quad \textbf{(3)}$$

Where Pi = preference of each alternative, fij = measure of the performance in the normalized matrix, wj = the weight of each criterion, m and n = the number of alternatives and the number of criteria. Accordingly, the normalized objective function for the proposed optimization method can be expressed with Equation 4.

$$\textbf{Normalized objective function} = \left(\frac{\textbf{Duration}*\mathbf{w_1}}{\sum_1^{\mathbf{m}} \textbf{Duration}}\right) + \left(\frac{\textbf{Cost}*\mathbf{w_2}}{\sum_1^{\mathbf{m}} \textbf{Cost}}\right) \quad \textbf{(4)}$$

3. Model Implementation

The developed method is applied to a numerical example that many researchers have used. Results from the performance of other methods [30, 31] were compared to examine the developed method's performance. The activity network and data of the project are shown in Figure 7 and Table 3, respectively. The indirect cost is \$2200 per day; the maximum labor capacity is ≤ 30 men; and the splitting cost is 100\$ per split. The delay penalty/opportunity cost was set to zero since the compared methods did not utilize it in their results. Furthermore, to illustrate the influence of the developed activity splitting function over the optimal project duration and cost and the final generation of the genetic algorithm optimization, the numerical example was solved under two different conditions.

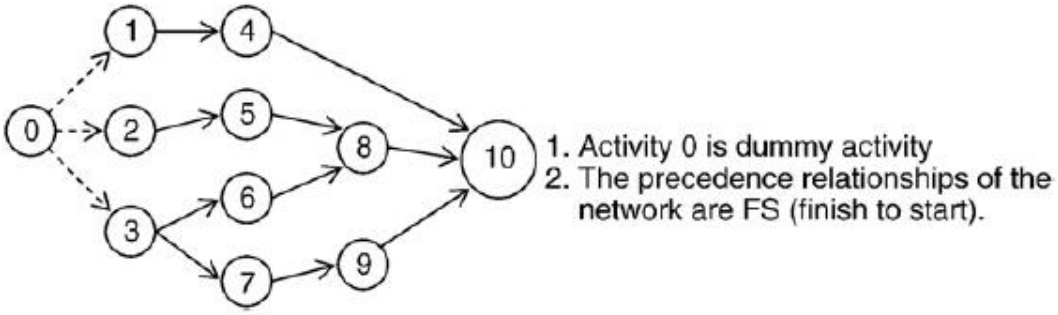

Figure 7. Numerical sample network (Chen and Weng, 2009).

Table 3. Activity execution modes (Chen and Weng, 2009).

Activity ID	Execution Mode	Duration	Cost	Resources requirements
1	1	5	7500	15
2	1	4	6400	16
	2	6	6000	10
	3	8	5600	7
	4	9	5400	6
3	1	6	7800	13
	2	8	7200	9
	3	10	7000	7
4	1	12	19200	16
	2	15	15000	10
	3	18	14400	8
5	1	22	39600	18
	2	24	38400	16
	3	26	36400	14
	4	28	33600	12
6	1	14	28000	20
	2	18	27000	15
	3	24	19200	8
7	1	9	15300	17
	2	10	14000	14
8	1	14	9800	7
	2	15	9000	6
	3	16	6400	4
9	1	15	7500	5
	2	18	7200	4
	3	20	6000	3
10	1	3	1200	4
	2	5	1000	2

3.1. Project scheduling with precedence relationships, resource constraints, and activity splitting

The NSGA-II set of non-dominated solutions after 100 generations with a population size of 50, mutation rate of 0.6, and a crossover rate of 0.5 is shown in Table 4. The sum weighted method was used to find the best solution among the set of non-dominated solutions. Since our optimization problem is a minimization case, the lowest preference index (PI) solution is the best alternative. It can be seen that the third solution is the best solution, with the lowest PI value of 0.1678564, a duration of 43 days, and a cost of $237,900. Simultaneously, the first solution is the most inadequate solution with the highest PI value of 0.24662072, a duration of 80 days, and a cost of $245,800. A breakdown of the project cost is summarized in Table 5.

Table 4. NSGA-II results with constraints and activity splitting.

Duration (Days)	Cost ($)	Normalized Duration	Normalized Cost	Preference Index (PI)
86	245800	0.3028169	0.19042454	0.24662072
50	245300	0.17605634	0.19003719	0.18304676
43	**237900**	**0.1514085**	**0.1843043**	**0.1678564**
45	241600	0.1584507	0.18717075	0.17281073
60	320200	0.21126761	0.24806322	0.22966541

Table 5. Breakdown of project cost.

Project Cost	**$237,900**
Direct Cost	$94,600
Indirect cost	$142,300
Splitting cost	$1,000
Delay penalty/Oppurinity cost	$0

3.2. Project scheduling with precedence relationships and resource constraints

After running 100 generations of the NSGA-II optimization method using both constraints without the splitting function, the non-dominated set of solutions is shown in Table 6. Figure 8 shows the solutions for the 100 generations, where the black curve depicts the optimal time-cost trade-off curve with the non-dominated set of solutions. From Table 6, it is clear that the fifth solution is the best one since it has the lowest PI value of 0.19282, a duration of 55 days, and a cost of $268,117. While comparing the two scenarios' results, it was evident that the project duration and cost increased by 27.9 % and 12.7 %, respectively. This can be explained by the fact that the splitting function was not applied in this scenario.

Table 6. NSGA-II results with constraints (no activity splitting).

Duration (Days)	Cost ($)	Normalized Duration	Normalized Cost	Preference Index (PI)
43	404598	0.16996047	0.25390604	0.21193326
48	330376	0.18972332	0.20732792	0.19852562
53	301979	0.2094862	0.1895073	0.1994968
54	288425	0.21343874	0.18100151	0.19722012
55	**268117**	**0.2173913**	**0.1682572**	**0.19282425**

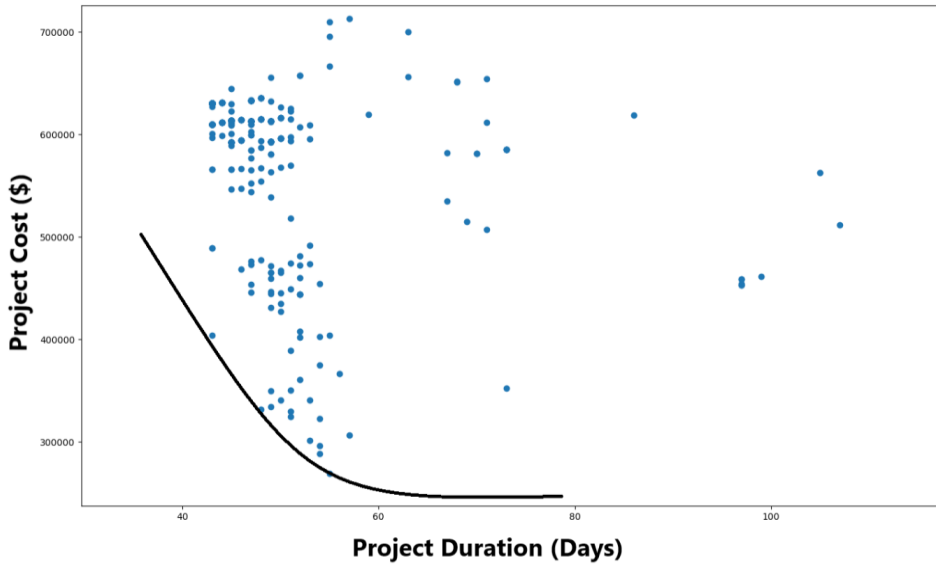

Figure 8. NSGA-II pareto front with constraints without activity splitting.

3.3. Comparison with other developed methods

Table 7 shows the applied developed method's results to the numerical example along with the other methods' results. The results illustrate the dominance of the developed method with its splitting function over the other methods. As seen in Table7, the developed method proved its efficiency in generating a better global optimum solution (duration and cost) and avoiding local minima entrapment. The proposed method's key advantage is its capability in splitting non-critical activities such that the resource constraint is considered, and the project is executed at the least possible duration and cost. The proposed method took approximately 5 min to solve the numerical example, with ten activities on a laptop machine having a 2.6 GHz processor speed. The developed method's processing time was higher than that of the heuristic method, developed by Hegazy and Menesi [31], and lower than that of the developed model by Chen and Weng [30]. The anticipated reason behind higher processing time than that of the heuristic method is that the developed method searches the possible solutions extensively before converging to a local optimum. Accordingly, it increases the probability of finding a near-optimum result.

Table 7. Comparison between the proposed method and other developed methods.

Research	Case study description	Results
Chen and Weng [30]	Ten activities having up to four discrete options that use varying amounts of one limited resource	Project cost = $244,000 Project duration = 56 days
Hegazy and Menesi [31]		Project cost = $245,900 Project duration = 59 days
Proposed method		Project cost = $237,900 Project duration = 43 days

4. Conclusion

This study introduced an integrated method for resource-constrained schedule compression that handles resource planning and project scheduling. The 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 project schedule optimization. The method evaluated the NSGA-II method for the resource-constrained scheduling optimization problem. The method was tested against other methods using a numerical example. It was observed that the developed method outperformed the previously developed methods, generating a better global optimum solution and avoiding entrapment in local minima during its reasonable processing time. Furthermore, the developed activity splitting tool proved its efficiency in obtaining a lower project duration and cost. Finally, it is anticipated that the developed method can help contractors generate an efficient construction schedule and speed up the project while ensuring efficient utilization of resources. The developed method can be further extended to consider the different types of resources. Finally, besides the genetic algorithm used in the presented method, other metaheuristic algorithms such as particle swarm optimization and ant colony optimization can also be evaluated to optimize the resource-constrained scheduling for activity crashing.

5. References

[1] Baloi D, and Price A D 2003 *International Journal of Project Management* **21** 261-9
[2] Memon A, Rahman I and Abdullah M 2010 *Factors Affecting Construction Cost in Mara Large Construction Project* 41-59
[3] Zheng D X M, Ng S T and Kumaraswamy M M 2005 *Journal of Construction Engineering and Management* **131** 81–91
[4] Golzarpoor B 2012 *(Doctoral dissertation) University of Waterloo* 76-91
[5] Aminbakhsh S and Sonmez R *2016 Expert Systems with Applications* **51** 177-85
[6] Toğan V and Eirgash M A 2018 *KSCE Journal of Civil Engineering* **23** 10-20
[7] Kaiafa S and Chassiakos A P 2015 *Procedia Engineering* **123** 260-7
[8] Hartmann S and Briskorn D 2010 *European Journal of Operational Research* **207** 1–14
[9] Elmaghraby S E 1977 *Activity networks: Project planning and control by network models* 45-76
[10] Slowinski R 1980 *The Journal of the Operational Research Society* **31** 711
[11] Węglarz J, Józefowska J, Mika M and Waligóra G 2011 *European Journal of Operational Research* **208** 177-205
[12] Al-Sakkaf A, Zayed T, Bagchi A, Mahmoud S and Pickup D 2020 *Smart and Sustainable Built Environment*.
[13] Li H and Zhang H 2013 *Automation in Construction* **35** 431-8
[14] Azizoglu M, Çetinkaya F C and Pamir S K 2015 *European J. of Industrial Engineering* **9** 450
[15] Altintas C and Azizoglu M 2020 *International Journal of Information Technology Project Management* **11** 55-70
[16] Chaleshtarti A S and Shadrokh S 2014 *Arabian Journal for Science and Engineering* **39** 8359-69
[17] Buddhakulsomsiri J and Kim D S 2006 *European Journal of Operational Research* **175** 279-95
[18] Al-Sakkaf A, Zayed T and Bagchi A 2020 *International Journal of Energy Optimization and Engineering (IJEOE)* **9** 49-73
[19] Peteghem V V and Vanhoucke M 2010 *European Journal of Operational Research* **201** 409-18
[20] Moukrim A, Quilliot A and Toussaint H 2015 *European Journal of Operational Research* **244** 360-68
[21] Afshar-Nadjafi B 2014 *Advances in Operations Research*, 1-10.
[22] Li F, Lai C and Shou Y 2011 *IEEE International Conference on Industrial Engineering and Engineering Management*, 45–79.
[23] Cheng J, Fowler J, Kempf K and Mason S 2015 *Computers & Operations Research*, **53** 275–287.
[24] Son J and Mattila K G 2004 *Journal of Construction Engineering and Management*, **130** (6), 887-894.
[25] Elmasry M, Zayed T Hawari A 2019 *Journal of Construction Engineering and Management,* **145** (2) 04018129.

[26] Elmenshawy M, and Marzouk M 2021 *Engineering, Construction and Architectural Management*, Vol. ahead-of-print No. ahead-of-print.
[27] Deb K, Pratap A, Agarwal S and Meyarivan T 2002 *IEEE Transactions on Evolutionary Computation*, **6** (2) 182-197.
[28] Khan A H 2017 *NSGA-II.py* (Version **1.0**) [Python].
[29] Abdelkader E M, Marzouk M and Zayed T 2019 *Soft Computing*, **23** (22) 12063-12086.
[30] Chen P.-H and Weng H 2009 *Automation in Construction*, **18** (4) 485–498.
[31] Hegazy T and Menesi W 2012 *Journal of Construction Engineering and Management*, **138** (6) 688–696.