Sakarya University Journal of Science

ISSN : 2147-835X
Publisher : Sakarya University

Vol. 28, No. 5, 959-977, 2024
DOI: https://doi.org/10.16984/saufenbilder.1467829

Research Article

Cost Minimization with Project Crashing: Comparison of the Traditional Solution and Genetic Algorithm Approach

Semih Cağlayan[1*], Sadık Yiğit[2]

[1] Sakarya University of Applied Sciences, Dept. of Civil Engineering, Sakarya, Türkiye, semihcaglayan@subu.edu.tr
[2] Zurich University of Applied Sciences, Centre for Building Technologies and Processes, Winterthur, Switzerland, sadik.yigit@zhaw.ch
[*]Corresponding Author

ARTICLE INFO

Keywords:
Project crashing
Resource allocation
Project scheduling
Cost minimization
Metaheuristic algorithms
Genetic algorithm

Article History:
Received: 13.04.2024
Accepted: 10.09.2024
Online Available: 14.10.2024

ABSTRACT

Existence of delays and cost overruns frequently puts the project viability in jeopardy. The integrated nature of these threats brings forward project scheduling as the primary determinant of project management success. The quality of project scheduling depends highly on the way resources are assigned to activities. In the project management literature, the efficiency of resource allocation is examined closely by the phenomenon called project crashing. This study introduces traditional and genetic algorithm approaches for the project crashing events and explains their steps in achieving the most efficient resource allocation. Within this context, the project crashing event is visualized, the insights of alternative approaches are described, and their implementations are illustrated with a case study. Besides, the procedures required for adopting the genetic algorithm approach to a typical problem are expressed. The case study illustration reveals the advantages and disadvantages of the genetic algorithm approach over the traditional approach. It is observed that the genetic algorithm approach can reach the solution in a single phase while the traditional approach requires multiple phases. On the other hand, the genetic algorithm approach may not reach the optimum solution unless the toolbox options are appropriately selected. This study presents the contribution of operational research to the project management body of knowledge by demonstrating the applicability and efficiency of genetic algorithm in the project crashing events. Researchers and industry practitioners may benefit from the proposed approach by following the indicated procedures to incorporate genetic algorithm into optimization issues in different fields.

1. Introduction

The subject of project management has attracted the attention of practitioners and researchers from different disciplines such as management science, organization theory, operations management, and social psychology [1]. Even though many factors have been defined for the successful delivery of projects [2], it is an undeniable fact that many projects fail to perform as intended and experience delays and cost overruns [3, 4], which may put project viability at risk [5]. The integrated nature of delays and cost overruns implies that these two phenomena should be studied together [6]. Poor scheduling, therefore, has been frequently specified as the main reason behind the project management failures [7].

Management of large-scale projects necessitates coordination of many activities with different costs and durations [8]. Achievement of project success requires organizations to efficiently assign resources to these activities. The project scheduling optimization has a wide range of applications in the fields of construction,

Cite as: S. Çağlayan, S. Yiğit (2024). Cost Minimization with Project Crashing: Comparison of the Traditional Solution and Genetic Algorithm Approach, *Sakarya University Journal of Science*, 28(5), 959-977. https://doi.org/10.16984/saufenbilder.1467829

production planning, and manufacturing [9]. The issue lies at the heart of project management and has attracted great attention in academia [10]. Many researchers have attempted to find solutions to the issue since 1960s. The solutions have focused on creating a project schedule to minimize the project cost and duration [11].

Project crashing is a schedule compression technique that aims to minimize the total cost of the project. The technique is based on analyzing the cost and duration trade-offs to obtain the greatest compression in project schedule for the least incremental cost. The project duration is reduced by shortening the critical path, which corresponds to the longest sequence of activities. Even though shortening the critical path requires greater resource allocation to certain activities on the critical path (implying an increase in activity costs), reduction in the project duration can provide savings in penalty associated with the project delay. As long as the savings obtained outweigh the increase in activity costs, such an attempt helps the project managers decrease the total cost of the project.

Project crashing has been traditionally carried out with a multi-phase schedule compression method. In recent years, development of metaheuristic algorithms like the genetic algorithm has enabled researchers to identify the optimized solutions for such complicated issues. Several researchers have utilized metaheuristic algorithms to overcome the challenges encountered in such events [12]. This study makes a comparison between the traditional solution and genetic algorithm approach for the project crashing events. In this regard, the project crashing event is visualized; the project crashing concept is introduced; and a typical problem is represented in a table format. The steps followed by the traditional approach are presented and the procedures necessary to adopt the genetic algorithm approach are clarified. Implementation of these alternative approaches on a typical problem is illustrated with a case study. The advantages and disadvantages of genetic algorithm adoption over the traditional solution are observed.

The paper is organized as follows: Section 2 describes the genetic algorithm concept and summarizes the fields genetic algorithm have been used for. Section 3 explains the methodology; visualization of the project crashing event, description of the alternative approaches, and illustration with a case study. The steps followed by the traditional solution and genetic algorithm approach are explained, and advantages/disadvantages are discussed in Section 4. Finally, Section 5 presents the conclusion; summary of the results, contribution to the body of knowledge, and limitations of the study.

2. Research Background

2.1. The genetic algorithm concepts

Genetic algorithm is a well-known population-based metaheuristic algorithm inspired from the biological evolution process [13]. The algorithm was proposed by John Holland in 1970s and is based on the Darwinian theory of the survival of the fittest [14]. The optimization process of genetic algorithm is illustrated in Figure 1. The process starts with randomly generating an initial population. A certain number of individuals are created, where each individual represents a solution the problem [15]. A fitness function is used to evaluate the fitness of each individual [16]. The fitness value indicates the likelihood of each individual to survive and reproduce in the new population [17].

Generation of the initial population is followed by evolution toward better solutions by means of genetic operators: selection, cross-over, and mutation [18]. The selection operator chooses the fitter individuals in the current population as parents to give birth to individuals in the new population [19]. The cross-over operator takes the chosen individuals and generates the new individuals by crossing the genes between pairs of parents [20]. The mutation operator changes the genes in the new population to increase the diversity [21]. It helps the algorithm exploit all the search space [22]. In short, the cross-over operator provides better future generations, while the mutation operator prevents the algorithm from getting stuck at local optima [23].

Generation of the new population by subjecting the current population to the genetic operators is

an iterative process. The iteration is repeated until the number of iterations equals to a predefined value for maximum generations, which implies that the termination criterion is met [24]. Satisfaction of the criterion results in termination of the algorithm [25]. The best-so-far solution, the individual with the most satisfactory fitness value, is regarded as the optimal solution. The results obtained for the optimal solution are displayed.

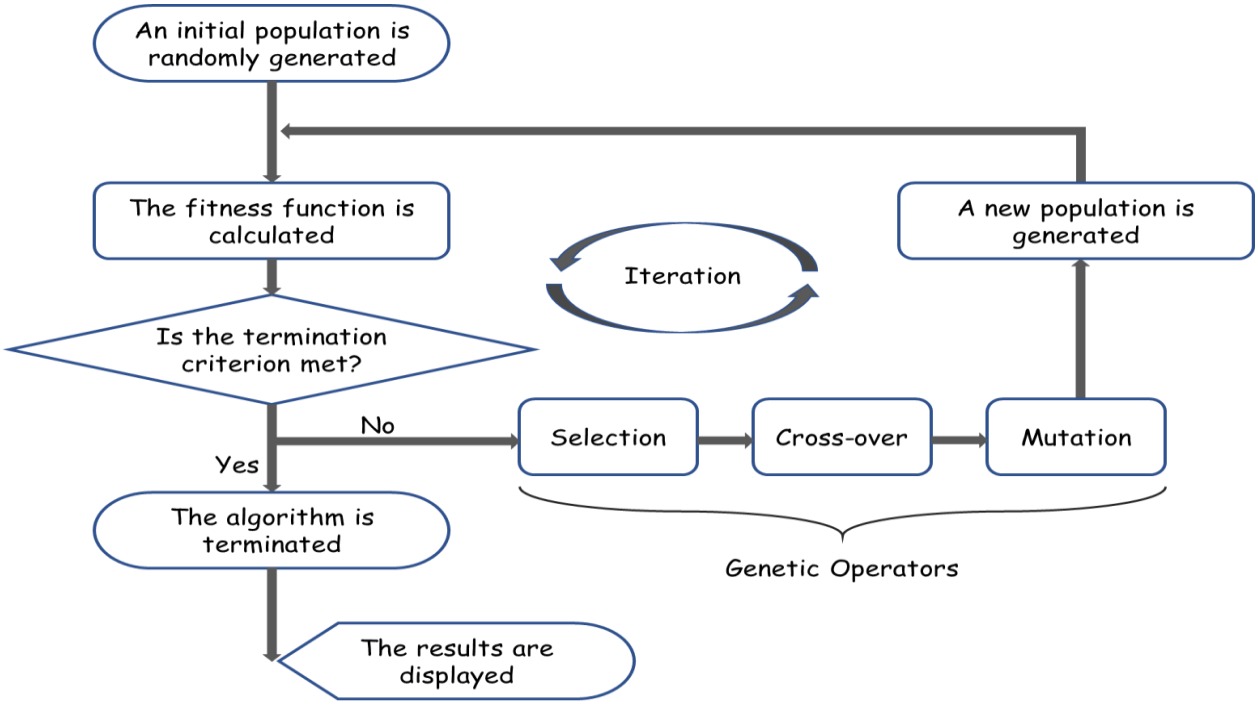

Figure 1. Genetic algorithm optimization process

2.2. Utilization of genetic algorithms for optimization

The metaheuristic algorithms, especially the genetic algorithms, have been utilized for various optimization purposes such as construction material design, energy efficiency in buildings, project layout planning, and project scheduling. A group of studies have attempted to optimize the design of construction elements. Genetic algorithms have been developed to optimize the design of multi-story composite steel frames [26], to minimize the construction cost of mass concrete [27], to produce asphalt mixtures complying with applicable specification requirements [28], to predict the adhesion strength [29], to mitigate the risk of early-age thermal cracking and delayed ettringite formation in mass concrete [30], to identify optimum parameter settings for aluminum-based hybrid metal matrix composite material [31]. Other metaheuristics-based algorithms have also been proposed for the design of cantilever retaining walls [32] and cantilever beams [33].

Another group of studies have used genetic algorithm to improve the energy efficiency in buildings. Several researchers have aimed at improving the energy efficiency through developing building retrofitting strategies [34]. Retrofitting has been done for school buildings [35], green building compliance [36], and residential buildings within a budget constraint [37]. Energy efficiency has been enhanced in another group of study with a focus on the heating energy consumption. The researchers have either developed algorithms to forecast the heating loads [38] or determined the optimum thermal design [39].

Several researchers have conducted studies for the optimization of project layout planning and logistics network design with the use of metaheuristic algorithms. Elements of the industrial production systems have been arranged in a number of studies as a part of the business operation strategy [40]. Multi-objective particle swarm optimization algorithm was proposed to minimize the construction safety risks of cranes and the total travelling distance of resources [41].

Non-dominated sorting genetic algorithm (NSGA) II was adapted to find the optimum location of facilities and transport network design to obtain sustainable supply-chain-network [42]. The link capacities and environmental protection were expressed as the problem constraints. A genetic algorithm-based model was proposed to solve the bus terminal location problem through finding efficient allocation patterns for assigning stations terminals [43]. The path planning in an unknown or partially known environment was resolved with artificial bee colony algorithm and evolutionary programming [44].

Project/program scheduling has been another application field of genetic algorithms. Algorithms have been used by several researchers for resource constraint optimization problems in construction [45, 46]. A genetic algorithm using a two-point crossover operator was proposed to a resource-constrained project scheduling problem with sequence dependent transfer times and the proposed algorithm could efficiently solve these problems [47]. A novel genetic algorithm incorporating complicated activity-dependencies for the resource leveling problem was developed by Li et al. [48]. The model could achieve near optimal solutions in fractions of a second. A NSGA-III-based algorithm was produced for a bi-objective hierarchical resource-constrained program scheduling problem [49]. The computational simulations confirmed the satisfactory performance of the model. In another study, a construction schedule was automatically generated by extracting data from a building information modeling product [50]. Behera and Sobhanayak [51] utilized genetic algorithm to optimize task scheduling in heterogeneous cloud computing environments.

Previous studies using the genetic algorithm concept for addressing certain optimization issues have mainly attempted to come up with solutions to these specific issues and have not clarified the steps that should be followed for repetition of the study in different fields. The literature obviously lacks an informative and explanatory study that visualizes a typical mathematical optimization problem and expresses the philosophy of utilizing the genetic algorithm to solve it. The procedures necessary for genetic algorithm adaptation need to be described so that the approach can also be adjusted for optimization issues in other fields and the use of genetic algorithm can be popularized.

This study aims to fill the gap in the literature by providing a detailed description for the use of genetic algorithm in project crashing events and demonstrating its advantages and disadvantages over the traditional solution through an illustrative case study. This study goes beyond optimization of a specific project scheduling issue and explains the philosophy behind integration of genetic algorithms into the project crashing events for the purpose of cost minimization. It clarifies the steps to be followed for proper integration of genetic algorithms into optimization problems.

3. Research Methodology

The flowchart of methodology is presented in Figure 2. It is composed of three main phases, namely (i) visualization of project crashing, (ii) description of alternative approaches, and (iii) case study illustration.

In the first phase, the project crashing event is introduced and visually represented. Categorization of the project cost is clarified, the project crashing concept is explained, and a typical problem is represented. The second phase presents the philosophy behind the traditional and proposed genetic algorithm approaches. It elaborates on the steps followed by the traditional and proposed genetic algorithm approaches to optimize resource allocation with the project crashing. The iterative process of the traditional approach is introduced and the requirements for adopting the genetic algorithm are expressed. The third phase involves implementation of the steps followed by these approaches on a case study.

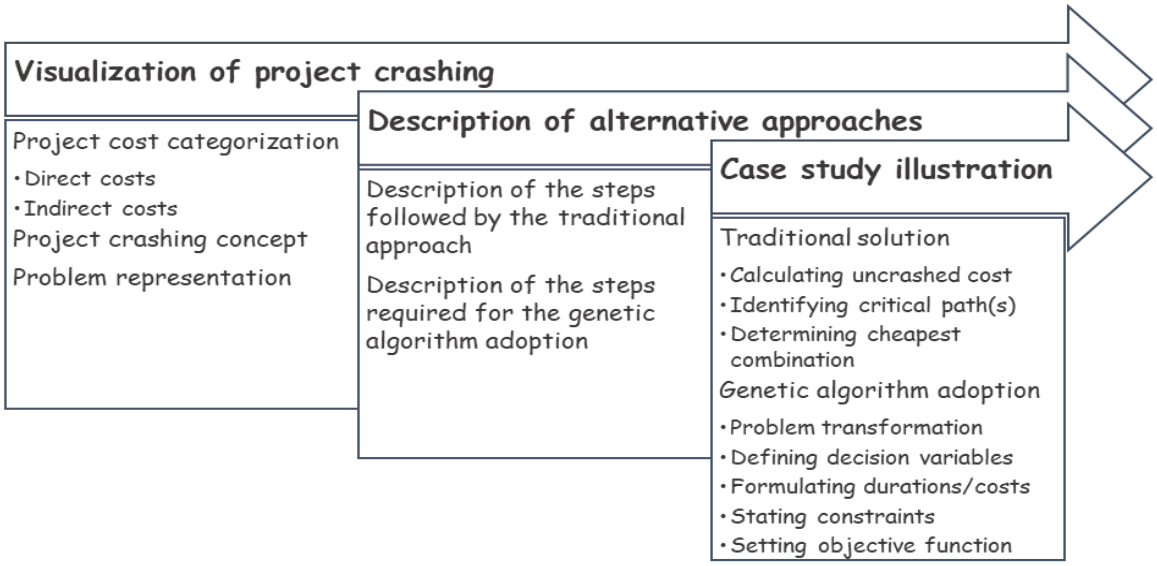

Figure 2. Flowchart of the methodology

3.1. Visualizing the project crashing events

The project cost is categorized into two different costs, namely the direct and indirect costs. The direct costs represent the costs directly attributable to the product. These may include the cost of raw materials, labor, and equipment costs. In project crashing events, the individual cost of each activity is evaluated under this category. In other words, the sum of the activity costs equals the direct costs. The indirect costs extend beyond the expenses realized in creating the product. General administration, supplies, and building rental costs can be given as the examples of indirect costs. Indirect costs are assumed to be linearly related to the project duration in the project crashing events.

Project crashing is a technique used to speed up the timeline of a project by providing more resources to certain activities in the critical path. More resources can be provided to the activities by assigning additional personnel or paying a premium. Shortening the project duration through allocating greater resources increases the direct cost. However, the indirect cost is decreased as it is directly associated with the duration of the project. Project crashing aims to shorten the project while keeping the project cost at a minimum. Determining the minimum project cost via the project crashing technique can be achieved through the traditional approach or the proposed genetic algorithm approach. These approaches follow different steps to calculate the optimum duration and cost of the project.

A typical project crashing problem is represented in Table 1. The first two columns show the list of activities and the relations between them. The third column indicates the normal duration (the duration when no crashing occurs) of each activity. The crushed duration stands for the duration of each activity after all the possible crashes are applied. Thus, the difference between the normal and crashed duration implies the number of crashes applicable to the corresponding activity.

To illustrate, if the difference is equal to two, the activity cannot be crashed more than two times (the duration cannot be decreased by more than two days). The remaining columns are the normal costs and the cost of each crash. The direct cost of the project is equal to the sum of the normal costs and costs of the crashes realized. The indirect cost is obtained by multiplying the overhead cost by the project duration determined by the critical path method (CPM).

3.2. Alternative approaches for project crashing

The steps followed by the two alternative approaches to minimize cost with project crashing are presented in Figure 3. In the traditional approach, it is firstly assumed that all the activities are realized in their normal durations/costs and the project duration/cost is calculated accordingly. CPM is used to calculate the project duration and identify the critical

Table 1. Representation of a typical project crashing problem

Activity	Predecessor	Normal Duration (days)	Crashed Duration (days)	Normal Cost ($)	Cost of Crashes ($)		
					1st Crash	2nd Crash	3rd Crash
A	…	…	…	…	…	…	…
B	…	…	…	…	…	…	…
C	…	…	…	…	…	…	…
D	…	…	…	…	…	…	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

*Overhead cost: … $/day

path(s). Acquisition of saving in indirect costs requires the project duration to be decreased by one unit. Therefore, the cheapest combination of activities is determined such that crushing these activities can satisfy the specified condition.

Shortening the project duration by one unit requires crashing all the critical paths. It is also necessary to explain the difference between crashing an activity and crashing a critical path. While the former implies decreasing the duration of the activity by one unit, the latter means crashing at least one of the activities in the corresponding path. The cost of crashing the selected activities is compared with the saving obtained from indirect costs. If the saving exceeds the cost of crashing; the selected activities are crashed, the new project duration and cost are calculated, and the process is repeated. Otherwise, the current project duration and cost are regarded as the optimum.

The proposed genetic algorithm approach follows totally different steps. Implementation of the proposed genetic algorithm approach requires transformation of the problem (Table 2). The duration/cost alternatives are presented for each activity rather than expressing the costs of crashes. Such a transformation is required for the formulations to be described in the following steps. A decision variable is defined for each activity as the number of times the corresponding activity is crashed. Afterwards, the duration and cost of each activity is expressed as a function of the decision variable. The number of times each activity can be crashed becomes the constraints for the decision variables. The objective function is defined as the minimization of the project cost. The project cost is formulated such that it becomes a function of the decision variables.

3.3. Case study illustration

The steps followed by two different approaches for the project crashing events are illustrated in a case study. A typical project crashing problem is presented in Table 3. The problem includes a total of 12 activities. The relations between the activities are given in the second column. A finish-to-start relationship exists between them. The normal and crashed duration of each activity are shown in the third and fourth columns.

For activity 1, the normal and crashed durations are 8 and 5 days, respectively. It is implied that the activity has the potential to be crashed for three days (8 days – 5 days). The fifth column shows the cost of the activity when it is completed in its normal duration. The last three columns show the cost of crashes. It is realized that the cost of crashing increases in each additional crash.

3.3.1. The traditional approach

The first step in the traditional approach is to calculate the project cost based on the normal durations and costs of activities. As already mentioned, the project cost is composed of the direct and indirect costs. The direct cost is calculated by summing up the costs stated in the fifth column of Table 3, which is determined as

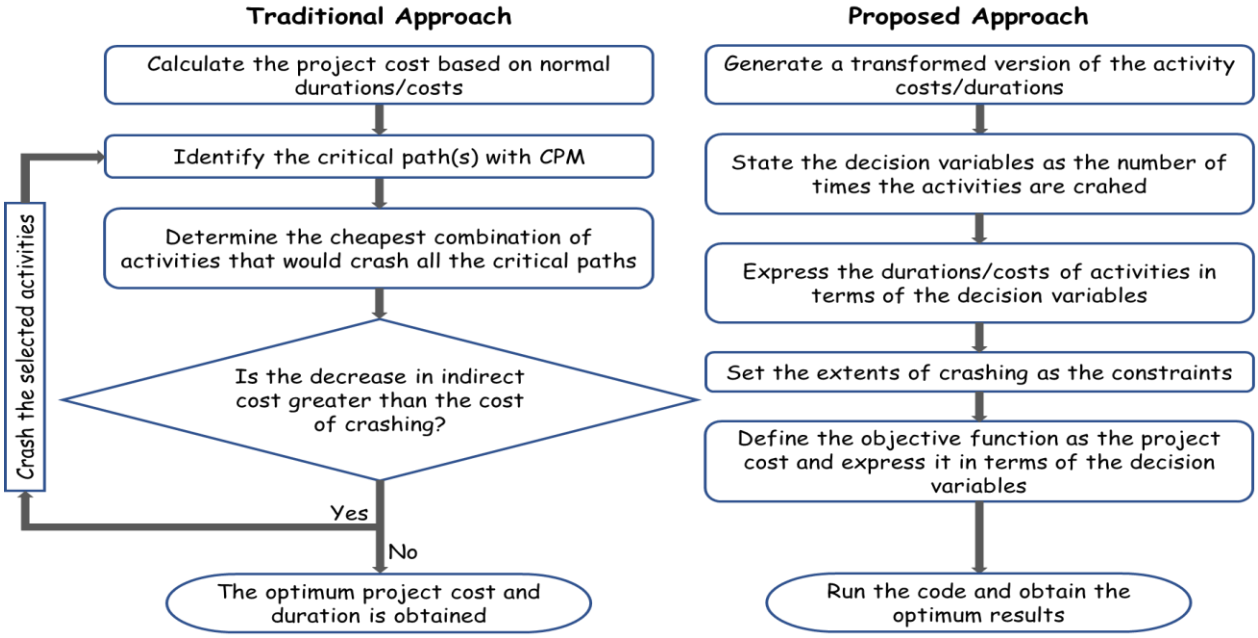

Figure 3. Steps followed by alternative approaches for project crashing events

Table 2. Representation of the transformed version of the problem

Activity	Predecessor	Normal Duration / Cost (days / $)	1st crash Duration / Cost (days / $)	2nd crash Duration / Cost (days / $)	3rd crash Duration / Cost (days / $)
A	…	… / …	… / …	… / …	… / …
B	…	… / …	… / …	… / …	… / …
C	…	… / …	… / …	… / …	… / …
D	…	… / …	… / …	… / …	… / …
⋮	⋮	⋮	⋮	⋮	⋮

*Overhead cost: … $/day

$23750. The indirect cost is calculated by multiplying the overhead cost by the project duration determined according to the CPM (Figure 4). The project duration and cost are identified as 21 days and $25850 (direct and indirect costs), respectively. The critical path is Activity 3 - Activity 6 - Activity 9 - Activity 10. The situation before crashing is summarized as:

- Duration: 21 days
- Cost: 23750 + 21 * 100 = $25850
- Critical path(s):
  - Act. 3 – Act. 6 – Act. 9 – Act. 10

Completing the project one day earlier provides a $100 saving in indirect cost. Reducing the project duration by one day requires all the critical paths to be crashed. Therefore, the cheapest combination of activities crashing all the critical paths is determined and compared with the saving in indirect cost. It is noticed that crashing Activity 9 by 1 day is cheapest way to crash the identified critical path and it is less than the saving realized ($20 vs $100).

The activity is crashed, the resultant project duration and cost are calculated, and the new critical paths are determined (Figure 5). The situation after the first crash is summarized as:

- 1st crash: Crash Activity 9 by 1 day
- Resultant duration: 20 days
- Resultant cost: 2585 + 20 – 100 = $25770
- Resultant critical path(s):
  - Act. 3 – Act. 6 – Act. 9 – Act. 10
  - Act. 1 – Act. 4 – Act. 8

Two critical paths are identified at the end of the first crash. The cheapest combination of the activities that can crash these critical paths are determined as Activity 1 and Activity 3.

Table 3. Information about the activity costs/durations

Activity	Predecessor	Normal Duration (days)	Crashed Duration (days)	Normal Cost ($)	Cost of Crashes ($)		
					1st Crash	2nd Crash	3rd Crash
1	-	8	5	2500	20	25	35
2	-	6	4	1200	30	45	-
3	-	7	5	1450	30	35	-
4	1, 2, 3	6	4	1350	30	30	-
5	1, 2, 3	4	2	1250	20	30	-
6	2, 3	5	3	2300	40	55	-
7	3	4	2	2000	40	45	-
8	4, 5	6	5	2750	40	-	-
9	6, 7	5	2	1800	20	35	50
10	4, 5, 9	4	2	850	45	55	-
11	7	8	5	3800	50	50	50
12	3	10	7	2500	15	20	25

*Overhead cost: … $/day

Crashing these activities costs $50 ($20+$30), which is less than the additional saving to be realized. These activities are crashed and project duration is decreased by one day (Figure 6). The situation after the second crash is summarized as:

- 2nd crash: Crash Activity 1 and Activity 3 by 1 day
- Resultant duration: 19 days
- Resultant cost: 25770 + 20 + 30 – 100 = $25720
- Resultant critical path(s):
  - Act. 3 – Act. 6 – Act. 9 – Act. 10
  - Act. 1 – Act. 4 – Act. 8
  - Act. 2 – Act. 6 – Act. 9 – Act. 10

Realization of the second crash leads to an additional critical path. The number of critical paths is increased to three. The cheapest combination to crash these paths is determined as Activity 1 and Activity 9. It should be noted that Activity 1 and Activity 9 are already crashed for one day in the second and first crashes, respectively. Thus, the cost of crashing becomes $25 for Activity 1 and $35 for Activity 9. It is realized that these three critical paths can be crashed simultaneously by crashing two activities because Activity 9 is the mutual activity in the first and third critical paths. As the cost of crashing ($25+$35) is less than the additional saving, the activities are crashed (Figure 7). The situation after the third crash is:

- 3rd crash: crash Activity 1 and Activity 9 by 1 day
- Resultant duration: 18 days
- Resultant cost: 25720 + 25 + 35 – 100 = $25680
- Resultant critical path(s):
  - Act. 3 – Act. 6 – Act. 9 – Act. 10
  - Act. 1 – Act. 4 – Act. 8
  - Act. 2 – Act. 6 – Act. 9 – Act. 10
  - Act. 3 – Act. 4 – Act. 8
  - Act. 3 – Act. 7 – Act. 11

Two additional critical paths occur after the third crash, resulting in a total of five critical paths. The cheapest way to crash these critical paths is to crash Activity 2, Activity 3, and Activity 4. Activity 3 is already crashed for one day in the second crash, so the crashing cost becomes $35 for Activity 3. The cost of crashing ($30+$35+$30) is still less than the potential saving in indirect costs. Thus, the fourth crash is also realized (Figure 8). The situation after the fourth crash is summarized as:

- 4th crash: crash Activity 2, Activity 3, and Activity 4 by 1 day
- Resultant duration: 17 days
- Resultant cost: 25680 + 30 + 35 + 30 – 100 = $25675
- Resultant critical path(s):
  - Act. 3 – Act. 6 – Act. 9 – Act. 10

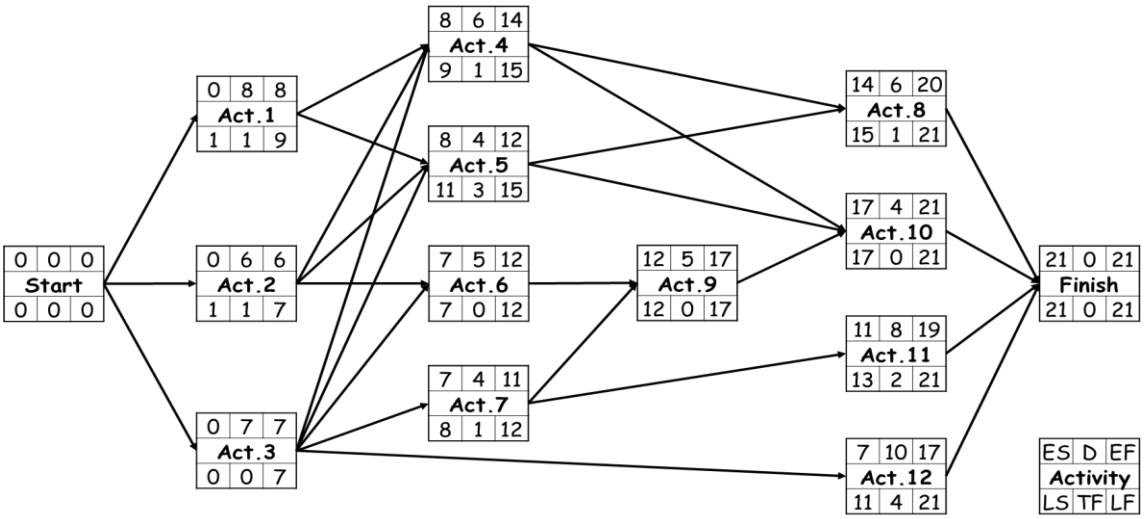

Figure 4. CPM analysis before crashing

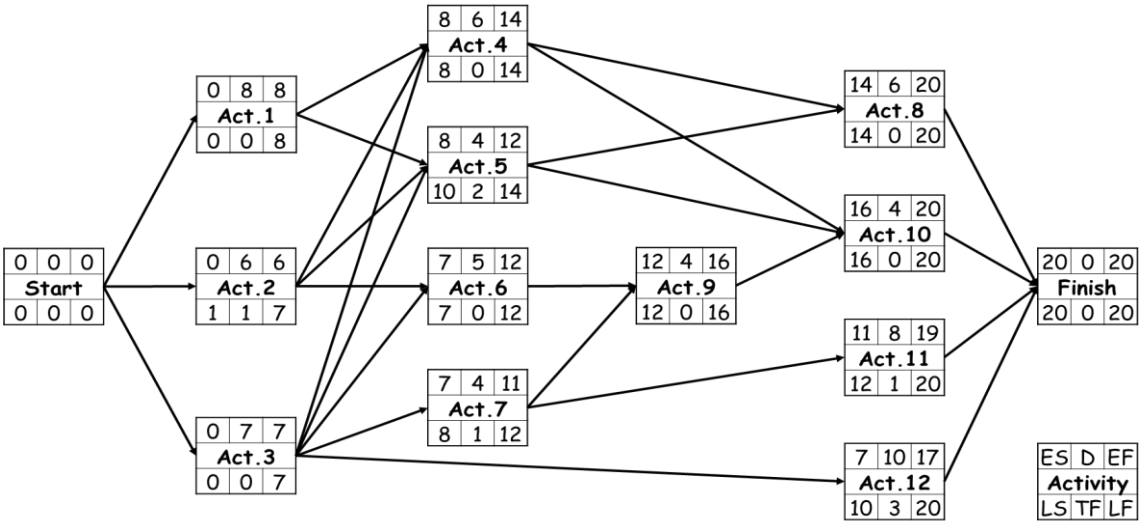

Figure 5. CPM analysis after 1st crash

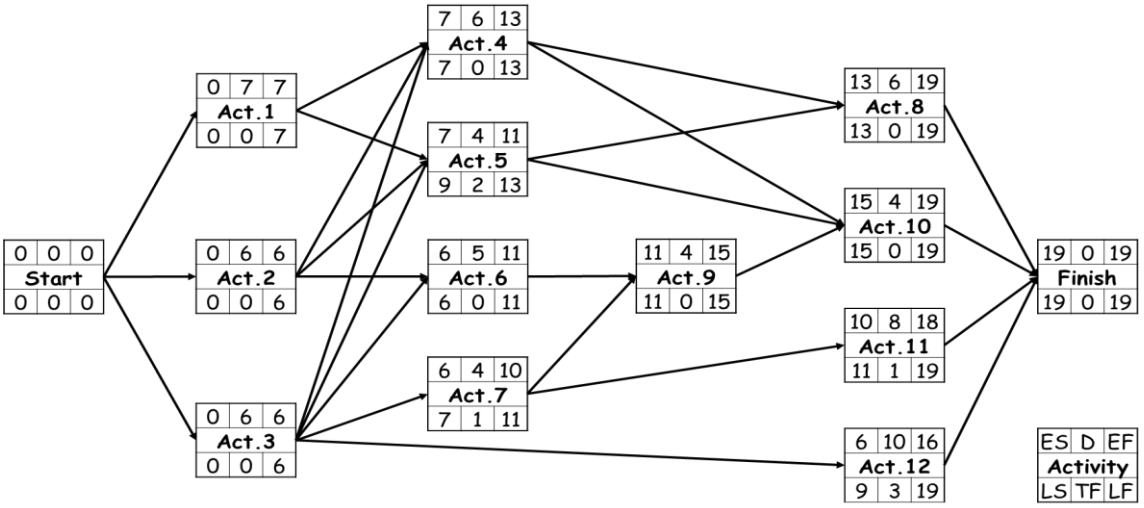

Figure 6. CPM analysis after 2nd crash

- Act. 1 – Act. 4 – Act. 8
- Act. 2 – Act. 6 – Act. 9 – Act. 10
- Act. 3 – Act. 7 – Act. 11

Four critical paths are identified after the fourth crash. The cheapest way to decrease project duration by one day is to crash Activity 4, Activity 6, and Activity 7. The cost of crashing

($30+$40+$40) exceeds the potential saving in indirect costs. It implies that crashing these activities results in an increase in total project cost. Therefore, the process is terminated and the current situation is described as the optimum. The optimum project duration and project cost are obtained as 17 days and $25675. The optimum situation is achieved by crashing Activity 1 by 2 days, Activity 2 by 1 day, Activity 3 by 2 days, Activity 4 by 1 day, and Activity 9 by 2 days. The other activities are realized in their normal durations and costs.

3.3.2. The genetic algorithm approach

The proposed genetic algorithm approach follows entirely different steps to optimize project crashing events. Adaptation of the genetic algorithm approach principally requires transformation of the problem. After the original problem is transformed, necessary coding is done in a programming platform that supports the genetic algorithm optimization. MATLAB has been preferred as a commercial programming platform. Definition of the decision variables, formulation of the durations/costs, statement of the constraints, and the expression of the objective function are explained to demonstrate the use of the genetic algorithm concept for the optimization of project crashing events.

Transforming the problem

A typical project crashing problem is most of the time presented as shown in Table 3. The normal duration/cost of each activity is given and costs of the crashes are indicated if available. The genetic algorithm approach requires a clear presentation of the alternatives for each activity. The cost and duration of an activity after each crash should be presented as a package. In this context, the presentation of the problem is transformed (Table 4). The alternatives for each activity are clearly observed in the transformed version. The number of alternatives for the activities ranges between two and four. Multiplication of the number of alternatives results in more than one million solutions. To be more precise, the genetic algorithm is expected to find the optimal solution among 1,119,744 solutions.

Defining the decision variables

The way the decision variables are defined is one of the most critical part of the genetic algorithm approach. The decision variable should be defined such that all the other variables and the objective function can be expressed as a function of the decision variable. For this purpose, the decision variable is defined as:

$$x_i = Number\ of\ times\ Activity\ i\ is\ crashed$$
$$x_i \in \mathbb{Z}^n$$

The decision variable is an integer that shows the number of times the corresponding activity is crashed. Thus, with this approach, the project crashing problem is transformed into a selection of how many times each activity should be crashed. The other variables that are to be expressed as a function of the decision variables are defined as follows:

$$d_i = Duration\ of\ Activity\ i$$
$$c_i = Cost\ of\ Activity\ i$$
$$ES_i = Early\ start\ time\ of\ Activity\ i$$
$$EF_i = Early\ finish\ time\ of\ Activity\ i$$

Formulating the durations/costs

It is already mentioned that the other variables and the objective function must be expressed as a function of the decision variables. The duration and cost of each activity are formulated as follows:

$$d_1\ and\ c_1 = \begin{cases} d_1 = 8\ days\ and\ c_1 = \$2500, & for\ x_1 = 0 \\ d_1 = 7\ days\ and\ c_1 = \$2520, & for\ x_1 = 1 \\ d_1 = 6\ days\ and\ c_1 = \$2545, & for\ x_1 = 2 \\ d_1 = 5\ days\ and\ c_1 = \$2580, & for\ x_1 = 3 \end{cases}$$

$$d_2\ and\ c_2 = \begin{cases} d_2 = 6\ days\ and\ c_2 = \$1200, & for\ x_2 = 0 \\ d_2 = 5\ days\ and\ c_2 = \$1230, & for\ x_2 = 1 \\ d_2 = 4\ days\ and\ c_2 = \$1275, & for\ x_2 = 2 \end{cases}$$

$$d_3\ and\ c_3 = \begin{cases} d_3 = 7\ days\ and\ c_3 = \$1450, & for\ x_3 = 0 \\ d_3 = 6\ days\ and\ c_3 = \$1480, & for\ x_3 = 1 \\ d_3 = 5\ days\ and\ c_3 = \$1515, & for\ x_3 = 2 \end{cases}$$

$$d_4\ and\ c_4 = \begin{cases} d_4 = 6\ days\ and\ c_4 = \$1350, & for\ x_4 = 0 \\ d_4 = 5\ days\ and\ c_4 = \$1380, & for\ x_4 = 1 \\ d_4 = 4\ days\ and\ c_4 = \$1410, & for\ x_4 = 2 \end{cases}$$

$$d_5 \text{ and } c_5 = \begin{cases} d_5 = 4 \text{ days and } c_5 = \$1250, & \text{for } x_5 = 0 \\ d_5 = 3 \text{ days and } c_5 = \$1270, & \text{for } x_5 = 1 \\ d_5 = 2 \text{ days and } c_5 = \$1300, & \text{for } x_5 = 2 \end{cases}$$

$$d_6 \text{ and } c_6 = \begin{cases} d_6 = 5 \text{ days and } c_6 = \$2300, & \text{for } x_6 = 0 \\ d_6 = 4 \text{ days and } c_6 = \$2340, & \text{for } x_6 = 1 \\ d_6 = 3 \text{ days and } c_6 = \$2395, & \text{for } x_6 = 2 \end{cases}$$

$$d_7 \text{ and } c_7 = \begin{cases} d_7 = 4 \text{ days and } c_7 = \$2000, & \text{for } x_7 = 0 \\ d_7 = 3 \text{ days and } c_7 = \$2040, & \text{for } x_7 = 1 \\ d_7 = 2 \text{ days and } c_7 = \$2085, & \text{for } x_7 = 2 \end{cases}$$

$$d_8 \text{ and } c_8 = \begin{cases} d_8 = 6 \text{ days and } c_8 = \$2750, & \text{for } x_8 = 0 \\ d_8 = 5 \text{ days and } c_8 = \$2790, & \text{for } x_8 = 1 \end{cases}$$

$$d_9 \text{ and } c_9 = \begin{cases} d_9 = 5 \text{ days and } c_9 = \$1800, & \text{for } x_9 = 0 \\ d_9 = 4 \text{ days and } c_9 = \$1820, & \text{for } x_9 = 1 \\ d_9 = 3 \text{ days and } c_9 = \$1855, & \text{for } x_9 = 2 \\ d_9 = 2 \text{ days and } c_9 = \$1905, & \text{for } x_9 = 3 \end{cases}$$

$$d_{10} \text{ and } c_{10} = \begin{cases} d_{10} = 4 \text{ days and } c_{10} = \$850, & \text{for } x_{10} = 0 \\ d_{10} = 3 \text{ days and } c_{10} = \$895, & \text{for } x_{10} = 1 \\ d_{10} = 2 \text{ days and } c_{10} = \$950, & \text{for } x_{10} = 2 \end{cases}$$

$$d_{11} \text{ and } c_{11} = \begin{cases} d_{11} = 8 \text{ days and } c_{11} = \$3800, & \text{for } x_{11} = 0 \\ d_{11} = 7 \text{ days and } c_{11} = \$3850, & \text{for } x_{11} = 1 \\ d_{11} = 6 \text{ days and } c_{11} = \$3900, & \text{for } x_{11} = 2 \\ d_{11} = 5 \text{ days and } c_{11} = \$3950, & \text{for } x_{11} = 3 \end{cases}$$

$$d_{12} \text{ and } c_{12} = \begin{cases} d_{12} = 10 \text{ days and } c_{12} = \$2500, & \text{for } x_{12} = 0 \\ d_{12} = 9 \text{ days and } c_{12} = \$2515, & \text{for } x_{12} = 1 \\ d_{12} = 8 \text{ days and } c_{12} = \$2535, & \text{for } x_{12} = 2 \\ d_{12} = 7 \text{ days and } c_{12} = \$2560, & \text{for } x_{12} = 3 \end{cases}$$

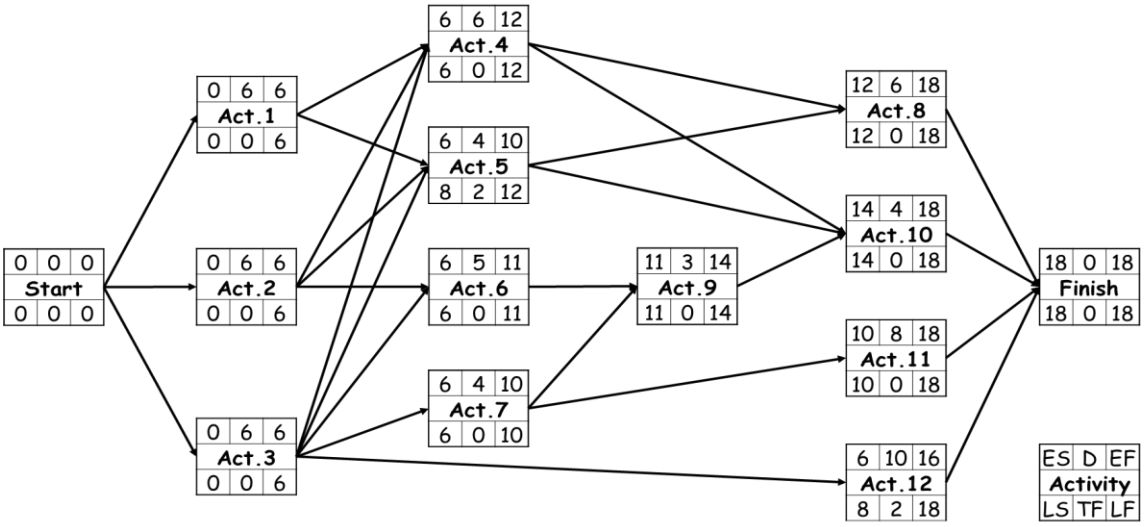

Figure 7. CPM analysis after 3rd crash

Stating the constraints

Another significant part of the optimization problems is the statement of the constraints. The constraints are the limitations the variables are subjected to. The constraints in project crashing events are about the number of times an activity can be crashed. An activity can be subjected to crashing for a minimum of zero times (corresponding to normal duration and cost) and for a maximum of 1-3 times as follows:

$$0 \le x_1 \le 3$$
$$0 \le x_2 \le 2$$
$$0 \le x_3 \le 2$$
$$0 \le x_4 \le 2$$
$$0 \le x_5 \le 2$$
$$0 \le x_6 \le 2$$
$$0 \le x_7 \le 2$$
$$0 \le x_8 \le 1$$
$$0 \le x_9 \le 3$$
$$0 \le x_{10} \le 2$$
$$0 \le x_{11} \le 3$$
$$0 \le x_{12} \le 3$$

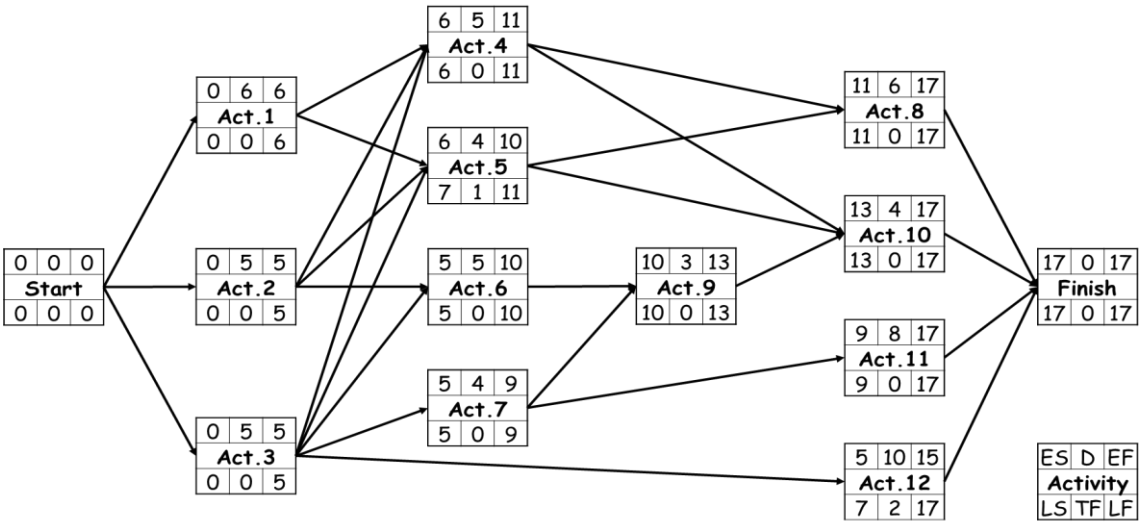

Figure 8. CPM analysis after 4th crash

Table 4. Transformed version of the activity costs/durations

Activity	Predecessor	Normal Duration / Cost (days / $)	1st crash Duration / Cost (days / $)	2nd crash Duration / Cost (days / $)	3rd crash Duration / Cost (days / $)
1	-	8 / 2500	7 / 2520	6 / 2545	5 / 2580
2	-	6 / 1200	5 / 1230	4 / 1275	-
3	-	7 / 1450	6 / 1480	5 / 1515	-
4	1, 2, 3	6 / 1350	5 / 1380	4 / 1410	-
5	1, 2, 3	4 / 1250	3 / 1270	2 / 1300	-
6	2, 3	5 / 2300	4 / 2340	3 / 2395	-
7	3	4 / 2000	3 / 2040	2 / 2085	-
8	4, 5	6 / 2750	5 / 2790	-	-
9	6, 7	5 / 1800	4 / 1820	3 / 1855	2 / 1905
10	4, 5, 9	4 / 850	3 / 895	2 / 950	-
11	7	8 / 3800	7 / 3850	6 / 3900	5 / 3950
12	3	10 / 2500	9 / 2515	8 / 2535	7 / 2560

*Overhead cost: 100 $/day

Setting the objective function

The final step of the genetic algorithm approach is to define and express the objective function. The aim of the project crashing event is to minimize the project cost ($c_{Project}$). As already mentioned, the project cost is equal to the sum of the direct and indirect costs. The direct costs are calculated by summing the individual cost of each activity ($c_1 + c_2 + c_3 \ldots + c_{12}$). The indirect costs are calculated by multiplying the overhead cost (100 $/day) by the project duration ($d_{Project}$). The objective function is formulated as follows:

$$c_{Project} = c_1 + c_2 + c_3 + c_4 + c_5 + c_6 + c_7 + c_8 + c_9 + c_{10} + c_{11} + c_{12} + 100 * d_{Project}$$

In this equation, the individual cost of each activity is a function of the decision variables and the overhead cost is 100 $/day. The project duration should also be expressed as a function of the decision variables. The project duration is calculated according to CPM as in the traditional approach. However, rather than calculating the project duration after each crash, the relations between the activities are formulated for once. The early start (ES) and early finish (EF) times of each activity are expressed as a function of the activity durations (d_i), which are already stated as a function of the decision variables (x_i). The project duration is equal to the EF of the last activity (Finish). The late start (LS) and late finish (LF) times of the activities are not formulated because there is no need to identify the total float (TF) times and the critical path(s) in the genetic algorithm approach. In other words, calculation of the project duration is sufficient. The project duration is formulated as follows:

$ES_{Start} = 0$
$EF_{Start} = 0$
$EF_i = ES_i + d_i, \quad for\ i = 1, 2, 3, \dots, 12$
$ES_1 = EF_{Start}$
$ES_2 = EF_{Start}$
$ES_3 = EF_{Start}$
$ES_4 = \max(EF_1, EF_2, EF_3)$
$ES_5 = \max(EF_1, EF_2, EF_3)$
$ES_6 = \max(EF_2, EF_3)$
$ES_7 = EF_3$
$ES_8 = \max(EF_4, EF_5)$
$ES_9 = \max(EF_6, EF_7)$
$ES_{10} = \max(EF_4, EF_5, EF_9)$
$ES_{11} = EF_7$
$ES_{12} = EF_3$
$ES_{Finish} = \max(EF_8, EF_{10}, EF_{11}, EF_{12})$
$EF_{Finish} = ES_{Finish}$
$d_{Project} = EF_{Finish}$

4. Research Results and Discussion

The steps indicated for adopting the genetic algorithm approach is coded in MATLAB and the results are obtained. The genetic algorithm toolbox options, which are essential for the optimization process are presented in Table 5. It should be noted that the options are highly dependent on experience and trial and error [52]. Arbitrarily estimated options may lead to convergence to a local optimum [53]. If the population size and maximum generations are insufficiently estimated, the algorithm may end up with a misleading solution. However, overestimation of these toolbox options can lead to an inefficient and time-consuming algorithm.

Table 5. Genetic algorithm options

Option	Description	Value
Population size	Number of individuals in the population	150
Selection	Selection of individuals for the next generation	Stochastic uniform
Maximum generations	Maximum number of iterations	100
Elitism	How many individuals in the current generation are guaranteed to survive	0.12 * 100
Tolerance function	Whether the average relative change in the best fitness function value is less than or equal to Funtool	10^{-8}
Cross-over function	The function that the algorithm uses to create cross-over members	Constraint dependent
Cross-over fraction	The fraction of the population created by the cross-over function	0.8
Mutation function	The function that produces mutation children	Constraint dependent

The genetic algorithm iterations are shown in Figure 9. It is noticed that the best members of the population do not show noticeable improvement after the 40th generation. It can be explained by the fact that the algorithm developed for the project crashing problem reaches nearly optimum values in 40 generations and stays there. Nevertheless, in an attempt to make sure that the optimum solution is obtained, the maximum number of iterations is selected as 100 (much greater than 40) as shown in Table 5. As the average change in the penalty fitness is lower than the tolerance function (10^{-8}), the process stops nearly at the 80th generation.

The values of the decision variables for the optimum solution are shown in Table 6. The optimum solution requires that Activity 1, Activity 2, Activity 3, Activity 4, and Activity 9 are crashed for 2, 1, 2, 1, and 2 times, respectively. Such a situation implies that greater resources are assigned to these activities. The other activities are realized in their normal durations and costs. The optimum cost of the project is determined as $25675. The solution is exactly the same as the solution in the traditional approach.

The solution of the project crashing problem is presented with two different approaches, namely the traditional approach and the proposed genetic algorithm approach. These two approaches arrive at the same solution by following entirely different steps. The traditional approach follows an iterative process that requires drawing the network diagram, identifying the critical path(s), and detecting the cheapest combination of activities to reduce the project duration by one day. These operations need to be realized before

each crash, which results in a multi-phase process.

The most challenging part of the traditional approach is the selection of cheapest combination of activities to crash. This part might be relatively easy in the first crashes as the network diagram results in a reasonable number of critical paths. However, as the process continues, the number of critical paths is increased and the selection becomes complicated. The critical paths and the number of crashing options before each crash are summarized in Table 7. It is seen that the number of critical paths is less than three before the first and second crashes. However, it is increased up to five in the following crashes. The activities in the critical paths are the candidates for selection. Each activity can either be "crashed" or "not crashed". Thus, there are 24, 27, 28, 210, and 210 options before the first, second, third, fourth, and fifth crashes, respectively. It is necessary to select the cheapest option (among more than a thousand options before the fourth and fifth crashes) that can crash all the critical paths.

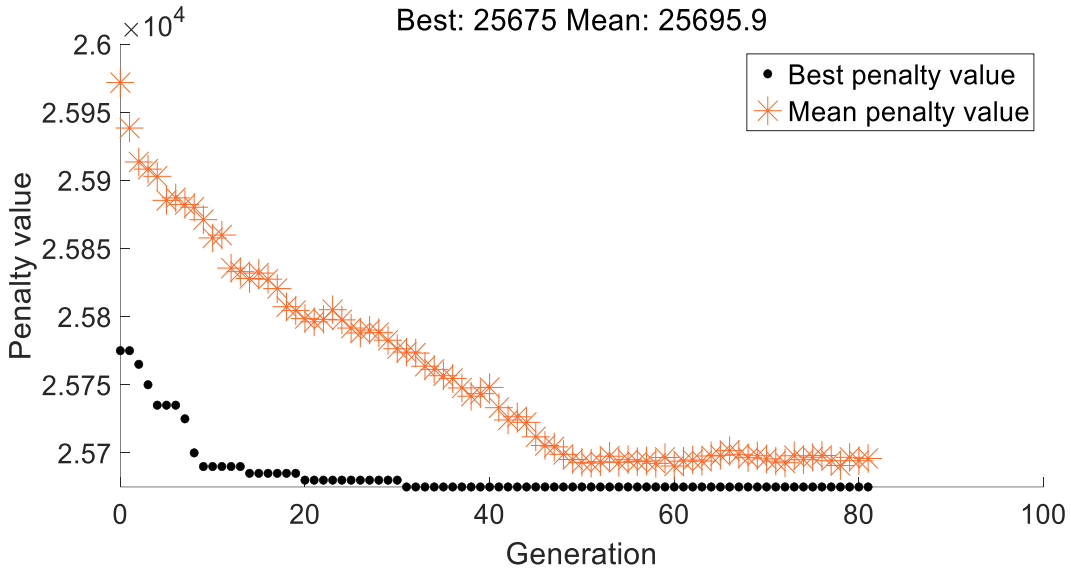

Figure 9. Genetic algorithm iterations

Table 6. The values of the decision variables at optimality

#	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
Result	2	1	2	1	0	0	0	0	2	0	0	0

Table 7. Number of crashing options in the traditional approach

Crash No	Critical Path(s)	Activities Concerned	# of Options
1	3 – 6 – 9 – 10	3, 6, 9, 10	16
2	3 – 6 – 9 – 10 1 – 4 – 8	1, 3, 4, 6, 8, 9, 10	128
3	3 – 6 – 9 – 10 1 – 4 – 8 2 – 6 – 9 – 10	1, 2, 3, 4, 6, 8, 9, 10	256
4	3 – 6 – 9 – 10 1 – 4 – 8 2 – 6 – 9 – 10 3 – 4 – 8 3 – 7 – 11	1, 2, 3, 4, 6, 7, 8, 9, 10, 11	1024
5	3 – 6 – 9 – 10 1 – 4 – 8 2 – 6 – 9 – 10 3 – 7 – 11	1, 2, 3, 4, 6, 7, 8, 9, 10, 11	1024

5. Conclusion

Project managers quite often suffer from delays and cost overruns in consequence of poor project scheduling. An efficient resource allocation process is essential for achieving the project success. It is a complicated process valid for many industries including construction and manufacturing. Numerous studies have been conducted in an attempt to create schedules minimizing the cost and duration. This study has proposed a genetic algorithm approach to achieve the most efficient resource allocation and minimize the total cost of the project with project crashing. The project crashing event has been visualized, alternative approaches have been described, and their implementations have been illustrated with a case study.

The results show that the same solution can be obtained with both approaches that follow entirely different steps. The main advantage of the genetic algorithm approach is that the solution is obtained in a single phase. Once the problem is appropriately transformed, decision variables are defined, and objective function is expressed in terms of the decision variables; the optimum solution can be easily obtained. The traditional approach is a multi-phase crashing process where a number of operations should be realized at each phase. On the other hand, it should be noted that the genetic algorithm approach may not reach optimum solution if the toolbox options are not appropriately selected. Selection of inappropriate toolbox options may lead to identification of the local optimum rather than the global optimum.

This study contributes to the body of knowledge by describing the philosophy behind the use of a genetic algorithm for cost minimization, demonstrating its applicability to the project crashing events, expressing the procedures required for its adaptation, and indicating the advantages and disadvantages compared to the traditional approach. It goes beyond optimizing a single phenomenon in the field of project management and enters into the spirit of integrating a genetic algorithm into any optimization event. It expresses how the problem should be handled, and data should be transformed to enable proper execution of the optimization process with genetic algorithm. The proposed approach is intended to encourage project managers to create effective schedules and allocate resources efficiently in their projects by means of such contemporary algorithms. The genetic algorithm adaptation procedures are expected to promote research focusing on the integration of metaheuristic approaches into optimization events in various fields.

The main limitation of the study is that both the traditional and proposed genetic algorithm approaches assume deterministic activity durations and costs. In practice, projects may involve uncertainty in these parameters [54]. The results at optimality might be subjected to variations as these parameters are subjected to changes. Nevertheless, it must be noted that the project crashing events are most of the time expressed in this format in the project management literature and potential solutions are developed accordingly. In addition, the steps followed in these approaches have been illustrated with a theoretical case rather than a real case. Illustration with a real case can be more influential and preferred in further studies. Further studies may also concentrate on the integration of the other metaheuristic algorithms (ant colony optimization, simulated annealing, particle swarm optimization, etc.) into various optimization events in different fields through following the steps expressed for the adaptation.

Article Information Form

Funding
The author(s) have not received any financial support for the research, authorship, or publication of this study.

Authors' Contribution
The authors contributed equally to the study.

The Declaration of Conflict of Interest/ Common Interest
No conflict of interest or common interest has been declared by the authors.

The Declaration of Ethics Committee Approval
This study does not require ethics committee permission or any special permission.

The Declaration of Research and Publication Ethics

The authors of the paper declare that they comply with the scientific, ethical and quotation rules of SAUJS in all processes of the paper and that they do not make any falsification on the data collected. In addition, they declare that Sakarya University Journal of Science and its editorial board have no responsibility for any ethical violations that may be encountered, and that this study has not been evaluated in any academic publication environment other than Sakarya University Journal of Science.

References

[1] H. L. Chen, "Performance measurement and the prediction of capital project failure," International Journal of Project Management, vol. 33, no. 6, pp. 1393-1404, 2015.

[2] E. Durna, B. Ozorhon, S. Caglayan, "Identifying critical success factors of public private partnership projects in Türkiye," Sakarya University Journal of Science, vol. 28, no. 1, pp. 30-50, 2024.

[3] T. Huo, H. Ren, W. Cai, G. Q. Shen, B. Liu, M. Zhu, H. Wu, "Measurement and dependence analysis of cost overruns in megatransport infrastructure projects: Case study in Hong Kong," Journal of Construction Engineering and Management, vol. 144, no. 3, pp. 05018001, 2018.

[4] P. Ballesteros-Perez, E. Sanz-Ablanedo, R. Soetanto, M. C. González-Cruz, G. D. Larsen, A. Cerezo-Narváez, "Duration and cost variability of construction activities: An empirical study," Journal of Construction Engineering and Management, vol. 146, no. 1, pp. 04019093, 2020.

[5] C. Callegari, A. Szklo, R. Schaeffer, "Cost overruns and delays in energy megaprojects: How big is big enough?" Energy Policy, vol. 114, pp. 211-220, 2018.

[6] G. Heravi, M. Mohammadian, "Investigating cost overruns and delay in urban construction projects in Iran," International Journal of Construction Management, vol. 21, no. 9, pp. 958-968, 2021.

[7] S. Durdyev, "Review of construction journals on causes of project cost overruns," Engineering, Construction and Architectural Management, vol. 28, no. 4, pp. 1241-1260, 2020.

[8] S. Zareei, "Project scheduling for constructing biogas plant using critical path method," Renewable and Sustainable Energy Reviews, vol. 81, pp. 756-759, 2018.

[9] H. F. Rahman, R. K. Chakrabortty, M. J. Ryan, "Memetic algorithm for solving resource constrained project scheduling problems," Automation in Construction, vol. 111, pp. 103052, 2020.

[10] R. Pellerin, N. Perrier, F. Berthaut, "A survey of hybrid metaheuristics for the resource-constrained project scheduling problem," European Journal of Operational Research, vol. 280, no. 2, pp. 395-416, 2020.

[11] M. Á. Vega-Velázquez, A. García-Nájera, H. Cervantes, "A survey on the software project scheduling problem," International Journal of Production Economics, vol. 202, pp. 145-161, 2018.

[12] E. Osaba, E. Villar-Rodriguez, J. Del Ser, A. J. Nebro, D. Molina, A. LaTorre, P. N. Suganthan, C. A. C. Coello, F. Herrera, "A tutorial on the design, experimentation and application of metaheuristic algorithms to real-world optimization problems," Swarm and Evolutionary Computation, vol. 64, pp. 100888, 2021.

[13] S. Katoch, S. S. Chauhan, V. Kumar, "A review on genetic algorithm: past, present, and future," Multimedia Tools and Applications, vol. 80, no. 5, pp. 8091-8126, 2021.

[14] Y. Fang, S. T. Ng, "Genetic algorithm for determining the construction logistics of precast components," Engineering, Construction and Architectural Management, vol. 26, no. 10, pp. 2289-2306, 2019.

[15] K. Kim, J. Walewski, Y. K. Cho, "Multiobjective construction schedule optimization using modified niched pareto genetic algorithm," Journal of Management in Engineering, vol. 32, no. 2, pp. 04015038, 2016.

[16] S. Mirjalili, J. S. Dong, A. S. Sadiq, H. Faris, "Genetic algorithm: Theory, literature review, and application in image reconstruction," Nature-Inspired Optimizers, pp. 69-85, 2020.

[17] M. S. El-Abbasy, A. Elazouni, T. Zayed, "MOSCOPEA: Multi-objective construction scheduling optimization using elitist non-dominated sorting genetic algorithm," Automation in Construction, vol. 71, pp. 153-170, 2016.

[18] S. Sabharwal, P. Bansal, N. Mittal, S. Malik, "Construction of mixed covering arrays for pair-wise testing using probabilistic approach in genetic algorithm," Arabian Journal for Science and Engineering, vol. 41, no. 8, pp. 2821-2835, 2016.

[19] A. Elkelesh, M. Ebada, S. Cammerer, S. ten Brink, "Decoder-tailored polar code design using the genetic algorithm," IEEE Transactions on Communications, vol. 67, no. 7, pp. 4521-4534, 2019.

[20] Z. Tong, "A genetic algorithm approach to optimizing the distribution of buildings in urban green space," Automation in Construction, vol. 72, pp. 46-51, 2016.

[21] S. M. Lim, A. B. M. Sultan, M. N. Sulaiman, A. Mustapha, K. Y. Leong, "Crossover and mutation operators of genetic algorithms," International Journal of Machine Learning and Computing, vol. 7, no. 1, pp. 9-12, 2017.

[22] Z. Shen, A. Hassani, Q. Shi, "Multi-objective time-cost optimization using Cobb-Douglas production function and hybrid genetic algorithm," Journal of Civil Engineering and Management, vol. 22, no. 2, pp. 187-198, 2016.

[23] M. Mangal, J. C. Cheng, "Automated optimization of steel reinforcement in RC building frames using building information modeling and hybrid genetic algorithm," Automation in Construction, vol. 90, pp. 39-57, 2018.

[24] S. RazaviAlavi, S. AbouRizk, "Genetic algorithm–simulation framework for decision making in construction site layout planning," Journal of Construction Engineering and Management, vol. 143, no. 1, pp. 04016084, 2017.

[25] L. Zhang, T. N. Wong, "An object-coding genetic algorithm for integrated process planning and scheduling," European Journal of Operational Research, vol. 244, no. 2, pp. 434-444, 2015.

[26] M. Artar, A. Daloglu, "The optimization of multi-storey composite steel frames with genetic algorithm including dynamic constraints," Technical Journal, vol. 26, no. 2, pp. 7077-7098, 2015.

[27] M. Rita, E. Fairbairn, F. Ribeiro, H. Andrade, H. Barbosa, "Optimization of mass concrete construction using a twofold parallel genetic algorithm," Applied Sciences, vol. 8, no. 3, pp. 399, 2018.

[28] H. Sebaaly, S. Varma, J. W. Maina, "Optimizing asphalt mix design process using artificial neural network and genetic algorithm," Construction and Building Materials, vol. 168, pp. 660-670, 2018.

[29] E. Hazir, T. Ozcan, K. H. Koç, "Prediction of adhesion strength using extreme learning machine and support vector regression optimized with genetic algorithm," Arabian Journal for Science and Engineering, vol. 45, pp. 6985-7004, 2020.

[30] A. A. Chiniforush, M. Gharehchaei, A. A. Nezhad, A. Castel, F. Moghaddam, L. Keyte, D. Hocking, S. Foster, "Minimising risk of early-age thermal cracking and delayed ettringite formation in concrete–A hybrid numerical simulation and genetic algorithm mix optimisation approach," Construction and Building Materials, vol. 299, pp. 124280, 2021.

[31] A. Kumar, N. Grover, A. Manna, R. Kumar, J. S. Chohan, S. Singh, S. Singh, C. I. Pruncu, "Multi-objective optimization of WEDM of aluminum hybrid composites using AHP and genetic algorithm," Arabian Journal for Science and Engineering, vol. 47, no. 7, pp. 8031-8043, 2022.

[32] E. Uray, O. Tan, S. Carbas, I. H. Erkan, "Metaheuristics-based pre-design guide for cantilever retaining walls," Technical Journal, vol. 32, no. 4, pp. 10967-10993, 2021.

[33] A. M. Rayeni, H. G. Arab, M. R. Ghasemi, "An effective improved multi-objective evolutionary algorithm (IMOEA) for solving constraint civil engineering optimization problems," Technical Journal, vol. 32, no. 2, pp. 10645-10674, 2021.

[34] I. Costa-Carrapiço, R. Raslan, J. N. González, "A systematic review of genetic algorithm-based multi-objective optimisation for building retrofitting strategies towards energy efficiency," Energy and Buildings, vol. 210, pp. 109690, 2020.

[35] Q. Li, L. Zhang, L. Zhang, X. Wu, "Optimizing energy efficiency and thermal comfort in building green retrofit," Energy, vol. 237, pp. 121509, 2021.

[36] Y. Fan, X. Xia, "Energy-efficiency building retrofit planning for green building compliance," Building and Environment, vol. 136, pp. 312-321, 2018.

[37] Y. He, N. Liao, J. Bi, L. Guo, "Investment decision-making optimization of energy efficiency retrofit measures in multiple buildings under financing budgetary restraint," Journal of Cleaner Production, vol. 215, pp. 1078-1094, 2019.

[38] L. T. Le, H. Nguyen, J. Dou, J. Zhou, "A comparative study of PSO-ANN, GA-ANN, ICA-ANN, and ABC-ANN in estimating the heating load of buildings' energy efficiency for smart city planning," Applied Sciences, vol. 9, no. 13, pp. 2630, 2019.

[39] S. N. Al-Saadi, K. S. Al-Jabri, "Optimization of envelope design for housing in hot climates using a genetic algorithm (GA) computational approach," Journal of Building Engineering, vol. 32, pp. 101712, 2020.

[40] P. Pérez-Gosende, J. Mula, M. Díaz-Madroñero, "Facility layout planning. An extended literature review," International Journal of Production Research, vol. 59, no. 12, pp. 3777-3816, 2021.

[41] M. Oral, S. Bazaati, S. Aydinli, E. Oral, "Construction site layout planning: Application of multi-objective particle swarm optimization," Technical Journal, vol. 29, no. 6, pp. 8691-8713, 2018.

[42] M. A. Brahami, M. Dahane, M. Souier, "Sustainable capacitated facility location/network design problem: a Non-dominated Sorting Genetic Algorithm based multiobjective approach," Annals of Operations Research, vol. 311, pp. 821-852, 2020.

[43] A. Taghavi, R. Ghanbari, K. Ghorbani-Moghadam, A. Davoodi, A. Emrouznejad,

"A genetic algorithm for solving bus terminal location problem using data envelopment analysis with multi-objective programming," Annals of Operations Research, vol. 309, pp. 259-276, 2022.

[44] S. Kumar, A. Sikander, "Optimum mobile robot path planning using improved artificial bee colony algorithm and evolutionary programming," Arabian Journal for Science and Engineering, vol. 47, no. 3, pp. 3519-3539, 2022.

[45] A. Mahdavian, A. Shojaei, "Hybrid genetic algorithm and constraint-based simulation framework for building construction project planning and control," Journal of Construction Engineering and Management, vol. 146, no. 12, pp. 04020140, 2020.

[46] J. Liu, Y. Liu, Y. Shi, J. Li, "Solving resource-constrained project scheduling problem via genetic algorithm," Journal of Computing in Civil Engineering, vol. 34, no. 2, pp. 04019055, 2020.

[47] R. L. Kadri, F. F. Boctor, "An efficient genetic algorithm to solve the resource-constrained project scheduling problem with transfer times: The single mode case," European Journal of Operational Research, vol. 265, no. 2, pp. 454-462, 2018.

[48] H. Li, L. Xiong, Y. Liu, H. Li, "An effective genetic algorithm for the resource levelling problem with generalised precedence relations," International Journal of Production Research, vol. 56, no. 5, pp. 2054-2075, 2018.

[49] W. Peng, J. Zhang, L. Chen, "A bi-objective hierarchical program scheduling problem and its solution based on NSGA-III," Annals of Operations Research, vol. 308, no. 1, pp. 389-414, 2022.

[50] Z. Wu, G. Ma, "Automatic generation of BIM-based construction schedule: Combining an ontology constraint rule and a genetic algorithm," Engineering, Construction and Architectural Management, vol. 30, no. 10, pp. 5253-5279, 2023.

[51] I. Behera, S. Sobhanayak, "Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach," Journal of Parallel and Distributed Computing, vol. 183, pp. 104766, 2024.

[52] S. Caglayan, S. Yigit, B. Ozorhon, G. Ozcan-Deniz, "A genetic algorithm-based envelope design optimisation for residential buildings," Proceedings of the Institution of Civil Engineers - Engineering Sustainability, vol. 173, no. 6, pp. 280-290, 2020.

[53] H. G. Lee, C. Y. Yi, D. E. Lee, D. Arditi, "An advanced stochastic time-cost tradeoff analysis based on a CPM-guided genetic algorithm," Computer-Aided Civil and Infrastructure Engineering, vol. 30, no. 10, pp. 824-842, 2015.

[54] S. Tao, C. Wu, Z. Sheng, X. Wang, "Stochastic project scheduling with hierarchical alternatives," Applied Mathematical Modelling," vol. 58, pp. 181-202, 2018.