Computers, Materials & Continua
DOI:10.32604/cmc.2021.017710
Article

A Compromise Programming to Task Assignment Problem in Software Development Project

Ngo Tung Son[1,2,*], **Jafreezal Jaafar**[1], **Izzatdin Abdul Aziz**[1], **Bui Ngoc Anh**[2], **Hoang Duc Binh**[2] **and Muhammad Umar Aftab**[3]

[1]Department of Computer and Information Sciences, Universiti Teknologi Petronas, Seri Iskandar, 32610, Malaysia
[2]Department of Information and Communication Technology, FPT University, Hanoi, 100000, Vietnam
[3]Department of Computer Science, National University of Computer and Emerging Sciences, Chiniot-Faisalabad Campus, Chiniot, 35400, Pakistan.
[*]Corresponding Author: Ngo Tung Son. Email: sonnt69@fe.edu.vn
Received: 08 February 2021; Accepted: 29 April 2021

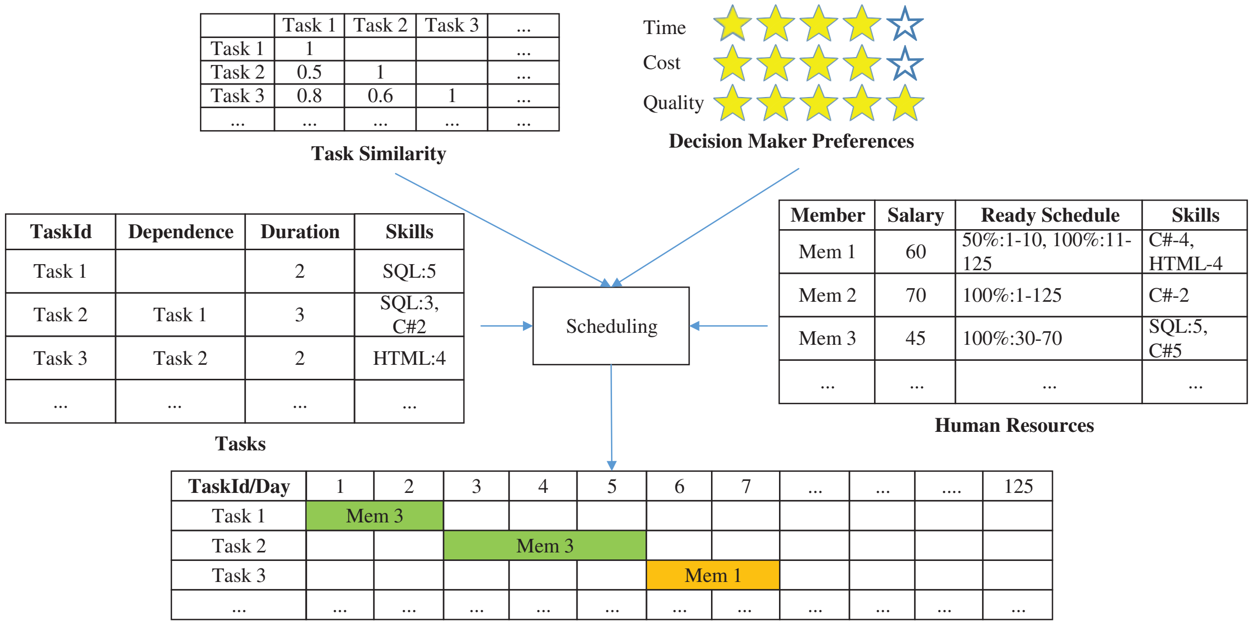
Abstract: The scheduling process that aims to assign tasks to members is a difficult job in project management. It plays a prerequisite role in determining the project's quality and sometimes winning the bidding process. This study aims to propose an approach based on multi-objective combinatorial optimization to do this automatically. The generated schedule directs the project to be completed with the shortest critical path, at the minimum cost, while maintaining its quality. There are several real-world business constraints related to human resources, the similarity of the tasks added to the optimization model, and the literature's traditional rules. To support the decision-maker to evaluate different decision strategies, we use compromise programming to transform multi-objective optimization (MOP) into a single-objective problem. We designed a genetic algorithm scheme to solve the transformed problem. The proposed method allows the incorporation of the model as a navigator for search agents in the optimal solution search process by transferring the objective function to the agents' fitness function. The optimizer can effectively find compromise solutions even if the user may or may not assign a priority to particular objectives. These are achieved through a combination of non-preference and preference approaches. The experimental results show that the proposed method worked well on the tested dataset.

Keywords: Makespan; RCPSP; scheduling; MOP; combinatorial optimization; compromise programming; genetic algorithm;

1 Introduction

1.1 Research Context

In software project management, scheduling plays a critical role in the success of a project. Scheduling starts with the work breakdown structure (WBS), which allows the whole project to be broken down into smaller tasks, which are assigned to project team members based on their skills and knowledge according to the working plan. In reality, projects run under constraints, including those of human resources, budget, and time. The lack of a helpful management toolset makes it difficult to achieve the goals of the shortest critical path, lowest cost, and highest quality. We develop an approach based on a multiobjective optimization model and a metaheuristic algorithm that generates an optimal project schedule to meet the above goals. Fig. 1 depicts the scheduler, which takes a list of tasks, project team members' profiles, and decision-makers' preferences as the model's inputs. The output is a detailed schedule. The time to complete a project depends on the available time, the similarity of tasks, and their interdependence. These factors also affect the cost and quality of the project.



	Task 1	Task 2	Task 3	...
Task 1	1			...
Task 2	0.5	1		...
Task 3	0.8	0.6	1	...
...	...	...	...	...

Task Similarity

Time	★★★★☆
Cost	★★★★☆
Quality	★★★★★

Decision Maker Preferences

TaskId	Dependence	Duration	Skills
Task 1		2	SQL:5
Task 2	Task 1	3	SQL:3, C#2
Task 3	Task 2	2	HTML:4
...	...	...	...

Tasks

Member	Salary	Ready Schedule	Skills
Mem 1	60	50%:1-10, 100%:11-125	C#-4, HTML-4
Mem 2	70	100%:1-125	C#-2
Mem 3	45	100%:30-70	SQL:5, C#5
...	...	...	...

Human Resources

TaskId/Day	1	2	3	4	5	6	7	...	...	....	125
Task 1	Mem 3										
Task 2			Mem 3								
Task 3						Mem 1					
...	...	...	...	...	...	...	...	...	...	...	...

Figure 1: Process of assigning tasks to each member of a project

1.2 Related Researches

Scheduling problems that deal with workforce constraints have the form of a resource-constrained project scheduling problem (RCPSP) [1–5]. Habibi et al. classified many RCPSP problem variations and mentioned modeling and algorithms to find optimal solutions for this type of problem in a 2018 survey [6]. Vanhoucke and Coelho presented an overview of state-of-the-art RCPSP algorithms [7]. The situation described in the first section of this paper is a multiobjective problem [8]. A primary objective is to solve the makespan problem, i.e., to complete the project in the shortest duration [9]. Makespan scheduling is described as follows: Assume M machines and N jobs for scheduling, where the n^{th} job takes $p_{n,m}$ units of time if scheduled on the m^{th} machine. Let J_m be the set of jobs on the m^{th} machine; then the machine's load is $l_m = \sum_{j \in J_m} p_{j,m}$. The maximum load is $l_{max} = \max_{m=1,\ldots,M} l_m$. Let $x_{n,m}$ be the decision variable to determine the n^{th} job assigned to the m^{th} machine. The minimum makespan becomes $\min(f)$, subject to: $\sum_{m=1}^{M} x_{n,m} = 1 \;\; \forall n = 1, 2, \ldots, N$, $\sum_{n=1}^{N} p_{n,m} * x_{n,m} \le f \;\; \forall m = 1, \ldots, M$, and $x_{n,m} \in \{0, 1\} \;\; \forall n = 1, \ldots, N, m = 1, \ldots, M$.

The interdependence of tasks often is expressed as a directed graph. Two fake tasks, the head (task 0) and tail (task N + 1), of zero duration, are added to formulate a single started/ended graph, as shown in Fig. 2. The makespan is now computed by any traversal method, such as breadth-first search (BFS). As mentioned in Section 1.1, the makespan may be affected in several ways. The tasks' interdependence determines the best critical path by the longest path in the graph. This constraint combines with the personnel's sequential workload, and their schedules may increase the critical path's length.

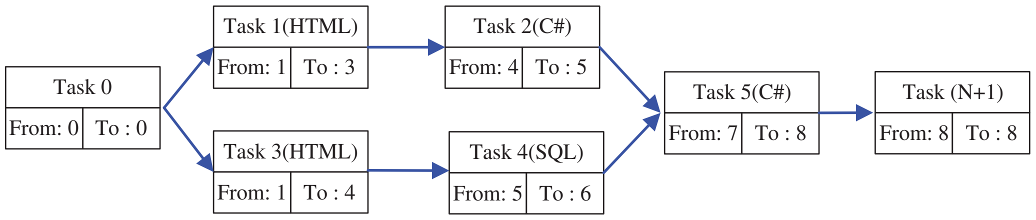


Figure 2: Head and tail tasks added to PERT chart to transform the problem to a single-start, single-end problem

Among recent studies on RCPSP and its variants Kumar et al. [10] proposed an algorithm to solve the classical RCPSP and minimize the makespan. Arian [11] designed an evolution algorithm to solve RCPSP, with the objective to minimize project cost, using binary decision variables to represent the time-unit when the activities started. Laurent et al. [12] and Stiti et al. [13] introduced a variant of multiside RCPSP, considering mobile and fixed resources in the transportation system. A variant of RCPSP subjected the project to renewable resources, each available for limited periods during the project life cycle, and applied a penalty cost for keeping resources for extra periods [14]. Quoc et al. [15] introduced Real-RCPSP, a variant of MS-RCPSP, and R-CSM, an algorithm to solve it based on cuckoo search. Hosseinian et al. [16] tried to maximize modularity to find high-quality employee communities and arrange them for tasks based on them. Nadjafi [17] proposed a mixed-integer programming formulation of RCPSP to minimize the total project cost, considering earliness-tardiness and preemption penalties. Ahmadpour et al. [18] presented a model of RCPSP by viewing a working calendar for project members and determining the skill factor of any member using the efficiency concept. Kellenbrink et al. [19] introduced two subproblems of RCPSP, the first to select activities, and the second to minimize makespan and total cost. Joshi et al. introduced a model to minimize the makespan or entire project duration. The objective was subject to activities' resource requirements at some finite integer value other than 1. The resources were staff members mastering one or more skills [20]. Tesch et al. [21] developed a mixed-integer programming formulation for RCPSP, improving the optimal problem-solving method over the previous approach by investigating relationships between event-based and time-indexed models.

Many other issues can arise in real-world situations, such as the division of labor in an organization, members' concentration in different project phases, and similarities of tasks. Project managers often assume that if a worker is assigned many similar tasks, then the job's cost will increase or decrease [22]. As in other engineering disciplines, to reuse previous work is critical to the progress and quality of software projects. In short, the more similar things a worker does the lower the cost. These aspects affect the objectives of project management.

RCPSP is combinatory optimization [23], and algorithms may not exist to find the optimal solution in polynomial time. In this case, metaheuristics have proved efficient in many case studies [24]. Tao et al. [25] introduced a procedure based on simulated annealing. Zhang et al. [26] designed a genetic algorithm with renewable resources and a single mode to perform each activity. Van et al. [27] introduced a genetic algorithm (GA) to solve both multimode and preemptive multimode RCPSP. However, most of these studies are in the form of problem-algorithm-result, and are not applicable to the proposed problem due to different business requirements. Based on results of previous studies, we construct a multiple-objective optimized mathematical model. Human resources, including skills, available time, salary, and task similarity, are constrained, in addition to constraints such as budget and deadline. We design a genetic algorithm to solve the model.

1.3 Contributions

In this research, we designed a new multi-objective optimization model for RCPSP problems that applies to software development in addition to our case study. It may be helpful in other task scheduling and project management work. Our model generates a detailed project schedule. The aim is to complete a project with the shortest duration while maintaining quality at a reasonable cost. It expresses the significant considerations of the decision-makers in the planning process. The model introduces some new rules related to human resources essential to software project management (and other engineering areas). We combine the preferred and non-preference approaches to transform the multi objective problem to a single-objective problem. Both methods have advantages and disadvantages and are suitable for different decision-makers. They are combined to leverage their benefits and cover each other's drawbacks. Section 2 of this paper describes the model.

The second contribution is to propose a new GA to solve the model. Unlike the usual mathematical programming or heuristic approach, the proposed method allows incorporation of the MOP approach in the algorithm design. To guide individuals in the search process, we use a distance-based objective function as their fitness. Section 3 describes the algorithm's implementation. The experimental procedure on the real dataset and its results are discussed in Section 4. We relate our conclusions in Section 5. Researchers in scheduling, planning, project administration, operations, and management can benefit from this research.

2 Multi-Objective Optimization Model

2.1 Problem Formulation

We define some frequently used variables:

- N is a number of tasks.
- d_n^{begin} and d_n^{end} are respectively the expected start and end dates of the n^{th} task.
- S is the number of required skills for the project.
- M is the number of candidates.
- $A = \left\{A_{i,j} \mid A_{i,j} \in \{0, 1\}\ i = 0, \ldots, N+1, j = 0, \ldots, (N+1)\right\}$ is the adjacency matrix of the directed-weighted graph that represents the dependences of the tasks. The 0^{th} and $(n+1)^{th}$ nodes are fake nodes added as the head and tail, where $d_0^{end} = d_0^{begin}$, $d_{n+1}^{end} = d_{n+1}^{begin}$.
- $K = \left\{K_{m,s} \mid K_{m,s} \in \{0, 1, \ldots, 5\}, m = 1, \ldots, M, s = 1, \ldots, S\right\}$, where $K_{m,s} \geq 1$ if the m^{th} candidate has the s^{th} skill; a greater value of $K_{m,s}$ indicates that the m^{th} member has more

experience with the s^{th} skill, and $K_{m,s} = 0$ means the m^{th} member has no experience with the s^{th} skill.

- $R = \left\{R_{n,s} \mid R_{n,s} \in \mathbb{R}^+, n = 1, \ldots, N, s = 1, \ldots, S\right\}$, where $R_{n,s}$ is the required level of the s^{th} skill. $R_{n,s}$ and $K_{m,s}$ are bounded by the same range of values.
- f_n^{begin} and f_n^{end} are the actual start and end times, respectively, of the n^{th} task when assigned to a member.
- T_m is the cost to hire the m^{th} member to work for one time unit.
- $Z = \left\{z_{i,j} \mid z_{i,j} \in [0, 1], i = 1, \ldots, N, j = 1, \ldots, N\right\}$ is the similarity matrix of the tasks, where $z_{i,j}$ is the similarity between the i^{th} and j^{th} tasks.
- D is the project deadline.
- $U = \left\{u_{m,\mathrm{d}} \mid u_{m,\mathrm{d}} \in [0, 1], u = 1, \ldots, M, d = 1, \ldots, D\right\}$ describes the concentration of personnel in a given time period; e.g., $u_{m,\mathrm{d}} = 0.5$ means the m^{th} member only attends 50% at time unit d.
- B is the project budget.
- Decision variables: $X = \left\{x_{n,m} \mid x_{n,m} \in \{0, 1\}, n = 1, \ldots, N, m = 1, \ldots, M\right\}$, where $x_{n,m} = 1$ if the n^{th} task is assigned to the m^{th} member, and $x_{n,m} = 0$ otherwise.

The project has three objectives:

- Minimize the critical path/makespan:

$$\min\left(g_1(X) = f_{n+1}^{end}\right) \tag{1}$$

Factors such as the concentration of members and similarity between tasks affect the time to complete the task.

- Minimize the cost of hiring team members:

$$\min\left(g_2(X) = \sum_{m=1}^{M}\left(\sum_{d=\min_{n=1,\ldots,N}\left(x_{n,m}*f_n^{begin}\right)}^{\max_{n=1,\ldots,N}\left(x_{n,m}*f_n^{end}\right)} (u_{m,\mathrm{d}} * T_m)\right)\right) \tag{2}$$

- Maximize the experience of selected candidates. Experience dramatically affects the quality of the project. The more experienced the candidate the less likely are errors:

$$max\left(\sum_{n=1}^{N}\sum_{m=1}^{M}\sum_{s=1}^{S} x_{n,m} * K_{m,s}\right) = \min(g_3(X) = -1 * \sum_{n=1}^{N}\sum_{m=1}^{M}\sum_{s=1}^{S} x_{n,m} * K_{m,s}), \tag{3}$$

subject to:

- All tasks are assigned to members:

$$\sum_{n=1}^{N}\sum_{m=1}^{M} x_{n,m} = N \tag{4}$$

- A task is assigned to only one member:

$$\sum_{\mathrm{m}=1}^{M} x_{n,m} = 1 \quad \forall n = 1, \ldots, N \tag{5}$$

- A member does not simultaneously perform two tasks:

$$\max\left(0, \min\left(x_{a,m} * f_a^{end}, x_{b,m} * f_b^{end}\right) - \max\left(x_{a,m} * f_a^{begin}, x_{b,m} * f_b^{begin}\right)\right) = 0$$

$$\forall \mathrm{m} = 1, \ldots, \mathrm{M}, \quad \mathrm{a} = 1, \ldots, (\mathrm{N} - 1), \quad \mathrm{b} = (\mathrm{a} + 1), \ldots, \mathrm{N} \tag{6}$$

- Dependent tasks cannot be executed at the same time:

$$A_{i,j} * f_j^{begin} - A_{i,j} * (f_i^{end} + 1) \geq 0 \quad \forall i = 1, \ldots, N - 1, \quad j = \mathrm{i} + 1, \ldots, N \tag{7}$$

- A task is assigned to the only member who qualifies:

$$R_{n,s} \leq x_{n,m} * K_{m,s} \quad \forall n = 1, \ldots, N, \quad m = 1, \ldots, M, \quad s = 1, \ldots, S \tag{8}$$

- The project ends on time:

$$f_{n+1}^{end} \leq D \tag{9}$$

- Costs do not exceed the budget:

$$\sum_{n=1}^{N} \sum_{m=1}^{M} \sum_{s=1}^{S} x_{n,m} * K_{m,s} \leq B \tag{10}$$

2.2 Approach for MOP

Ignizio said there is no one best approach to all types of multiobjective mathematical programming problems [28]. Solutions to multiobjective optimization problems (MOPs) are categorized as either preference or non-preference [29]. The first approach is to use preference information archived by interviewing the decision-maker to determine the solution that best satisfies these preferences. The second approach assumes that no decision-maker was available, and that we can identify a compromise solution without preference information. Decision-makers often cannot determine priorities, which leads them to try many parameters and select one of several solutions. It is challenging in practice to define meaningful ranges of parameters. The computational cost of a search algorithm is a barrier to finding appropriate parameter values to determine a final solution. Compromise programming allows a one-shot solution.

Compromise programming can solve the above problem by identifying an ideal solution [30] as a point of reference and finding a solution as close as possible to the ideal point. Ngo et al. [31] applied it to their MOP, introducing the concepts of "deep" and "wide" to access the skills of candidate teams. Xiong et al. [32] proposed two levels of compromise programming to allocate resources between pavement and bridge deck maintenance. Poff et al. [33] used compromise programming to represent 20 objective functions to determine the preferred decision variable value and assign a closeness level to an ideal solution. Ngo et al. [34] introduced CP as the difference between the perfect and actual points to solve a multiobjective timetabling problem, and tried to answer strategic questions as decision-makers in simple terms. In such a situation, we answer three questions, and we believe a decision-maker would do the same. What is the earliest time the project can be completed? The shorter the better. What is the lowest cost? The lower the better. At what level should the project quality be? The higher the better.

The answers above can represent the decision-maker's aspirations when unable to define any preferred objectives.

$E = \{E_i \mid i = 1, \ldots, 3\}$ denotes the expected point, where $E = \left[d_{n+1}^{end}, 0, \sum_{n=1}^{N} \sum_{m=1}^{M} \sum_{s=1}^{S} K_{m,s}\right]$

$O = \{O_i \mid i = 1, \ldots, 3\}$ represents the actual solution, where

$$O_i = \left[f_{n+1}^{end}, \sum_{m=1}^{M} \left(\sum_{d=\min_{n=1,\ldots,N}\left(x_{n,m} * f_n^{begin}\right)}^{\max_{n=1,\ldots,N}\left(x_{n,m} * f_n^{end}\right)} \left(u_{m,\mathrm{d}} * T_m\right) \right), \sum_{n=1}^{N} \sum_{m=1}^{M} \sum_{s=1}^{S} x_{n,m} * K_{m,s} \right]$$

Instead of maintaining the three goals of the original MOP, we have just the goal to find the point O closest to E. Compromise programming allows us to use any distance function. We select the Euclidean distance to build a new objective function,

$$minimize\,(distance\,(E, O)) = \sqrt{\sum_{i=1}^{3} (E_i - O_i)^2}$$

Many situations can occur during project planning. We need to assign a priority to each goal. The considered problem has three targets, which can be graphically visualized to better realize the effects of strategies. This is hard to achieve with compromise programming, although it is straightforward to explain that the decision-maker wants to balance goals. Another approach is to scalarize a MOP, i.e., to formulate a single-objective optimization problem whose solutions are Pareto optimal solutions to the MOP. The original multiobjective functions are scalarized and written as $min\left(\sum_{i=1}^{3} w_i * g_i(X)\right)$, where w_i is the weight of the i^{th} objective function. We want to reuse the good features of compromise programming and linear scalarizing, so we combine the approaches. The distance function's size is influenced by spatial dimensions [35], and can be manipulated using dimensional weights. The objective function becomes

$$minimize\,(distance\,(E, O)) = \sqrt{\sum_{i=1}^{3} w_i * norm((E_i - O_i)^2)} \tag{11}$$

where the *norm* function normalizes the features of the distance function to the same range.

3 Proposed Algorithm

3.1 Introduction to Genetic Algorithm

Evolutionary algorithms (EA) are mainstream stochastic methods for MOP [36]. In the family of EA, genetic algorithms (GA) are popular choices for engineering applications [37]. It starts searching for the optimal solutions in solution space by forming a population/set of possible solutions. Then new solutions are created by breeding the best fitness individual from the population to generate the new generation. Over the iterations, the population retains the best genetic hybridization combined with mutations to create a better next generation. GA gives a guideline for solving the problem. However, each issue needs a different scheme design. In this study, we introduce a version of the GA to solve the MOP model.

3.2 GA Scheme

Denote:

- U is population size.
- G denotes the convergence condition.
- $P^{(g)} = \{P_i^{(g)}, P_2^{(g)}, \ldots, P_U^{(g)}\}$ denotes population.
- $P_i^{(g)} = \left\{P_{i,1}^{(g)}, P_{i,2}^{(g)}, \ldots, P_{i,N}^{(g)}\right\}$ is an individual of the population.

where: $P_{i,e}^{(g)} = m \in M_e$

- M_e represents the set of members who able to do a task e^{th}.
- Our proposed genetic algorithm's scheme described as following:
- Initialize $g = 1$
- $timest_i$ is the time that a member i^{th} start the first task;
- $timefn_i$ is the time that a member i^{th} complete the last task;

The Genetic Algorithms expressed in 6 steps as follows:

Step 1: Randomly generate the $P_i^{(g)}\ \forall i = 1, 2, \ldots, U$.

Step 2: Compute fitness value corresponding to $P_i^{(g)}\ \forall i = 1, 2, \ldots, U$ as following:

Step 2.1: Update f_n^{start}, f_n^{end}, $timest_m$ and $timefn_m$ using BFS on matrix A and population $P_i^{(g)}$.

The needed duration to complete a task n^{th} affected by the task similarity, effort distribution of the assigned member. The f_n^{end} can be computed as:

$$f_n^{end} = f_n^{start} + actual\,(n, m)$$

where: $actual(n, m)$ is the estimated effort to complete a task n^{th} by member m^{th}. The *actual* defined as follow.

Function: *actual*

```
Input: $n, m$
1: $d = \left(\left(d_n^{end} - d_n^{begin}\right) + 1\right) * reduct\left(\max_{j=1,\ldots,N}\left(z_{n,j}\right)\right)$ Where $reduct$ is a fuzzy function, it's
   returned valueindicated by different project context.
2: $time := 1$
3: $count := 0$
4: While $time > 0$
5: begin
6:     $time := time - \frac{1}{d} * u_{m,f_n^{start}+count}$
7:     $count := count + 1$
8: end
9: return $count$
```

Step 2.2: Define *cost*, *Mtime*, *Mcost*, *Mquality* as following.

- $cost = \left(\sum_{m=1}^{M} T_m * (timefn_m - timest_m)\right)$
- $Mtime = \sum_{e=1}^{N} (d_e^{end} - d_e^{start})$
- $Mcost = \max_{e=1,\ldots,M} T_e * Mtime$
- $Mquality = \sum_{e=1}^{N} \sum_{x=1}^{S} \left(R_{e,x} * \max\left(K_{z,x}\right)\right) \ \forall z = 1, \ldots, M.$

Step 2.3: The fitness values $P_i^{(g)}.fitness$ could be biased due to the difference in ranges of its dimensions. We normalize features to the range [0, 1]. The distance function (11) is rewrite in form of fitness of the individual as:

$$P_i^{(g)}.fitness = \sqrt{w_1 * \left(\frac{f_{N+1}^{end} - d_{n+1}^{end}}{\text{Mtime}}\right)^2 + w_2 * \left(\frac{cost}{Mcost}\right)^2 + w_3 * \left(\frac{Mquality - \sum_{n=1}^{N}\sum_{m=1}^{M}\sum_{s=1}^{S} x_{n,m} * K_{m,s}}{Mquality}\right)^2}$$

Step 2.4: Sort the $P^{(g)}$ order by $P_i^{(g)}.fitness$ ascending.

Step 3: Elitism Selection: keep an individual $b = \underset{i=1,\ldots,U}{\text{argmin}} P_i^{(g)}.fitness$ that returns the best fitness for next-generation. Denote the best fitness value of the g^{th} generation as b^g. The first individual of the next generation created as: $P^{(g+1)}.add(P_{\text{b}}^{(g)})$.

Step 4: Crossover and mutation: Denote w as the mutation rate. Denote σ as tournament size selection.

For each $i = 2, \ldots, U$.

Step 4.1: Create 2 new populations with a size of σrespectively:

$$P^{(g),dad} = \left\{P_1^{(g),dad}, P_2^{(g),dad}, \ldots, P_\sigma^{(g),dad}\right\}$$

and

$$P^{(g),mom} = \left\{P_1^{(g),mom}, P_2^{(g),mom}, \ldots, P_\sigma^{(g),mom}\right\}$$

where:

$$P_i^{(g),dad} = rand\left(P^G\right) \quad \forall \ i = 1, \ldots, \sigma$$

$$P_i^{(g),mom} = rand\left(P^G\right) \quad \forall i = 1, \ldots, \sigma$$

$rand(P^G)$ is the function that return an item from the input list P^G. The returned item is not duplicated with the previous ones.

Step 4.2: Create best genes from $P^{(g),dad}$ and $P^{(g),mom}$ to generate the next generation.

$$d_{dad} = argmax_{P_i^{(g),dad} \in P^{(g),dad}} \left(P_i^{(g),dad}.fitness\right)$$

$$d_{mom} = argmax_{P_i^{(g),mom} \in P^{(g),mom}} \left(P_i^{(g),mom}.fitness\right)$$

Set $r = rand([0, 1])$

$$P_{i,j}^{(g+1)} = \begin{cases} rand\left(d_{dad,j}, d_{mom,j}\right) & if \;\; r > w \\ rand\,(j) & otherwise \end{cases} \qquad \forall j = 1, \ldots, N$$

where:

$rand([0, 1])$ returns a float number in range [0,1].

$rand\left(d_{dad,j}, d_{mom,j}\right)$returns $d_{dad,j}$ or $d_{dad,j}$ randomly.

$rand\,(j)$ randomly returns a value corresponding to a cell j^{th} of a choronosome.

Step 5: Constraint validation checking: if there is an individual $P_i^{(g+1)}$ that violates any constraints defined in Eqs. (4)–(10), then it is removed from the set of results.

Step 6: Return to Step 2 with $g = g + 1$ until $b^g = b^{g-1} = \ldots = b^{g-G}$.

4 Experiments

We validated the proposed method with the real datasets obtained after the WBS process of a software project. The results show that the algorithm works well with the tested dataset. It consisted of 417 dependent tasks, 19 candidates of project members, and 27 required skills. The algorithm was implemented in Java-8 and executed on a computer with an Intel Core I5-8250U CPU @1.60 GHz, with 4 GB RAM. It is essential to find appropriate values for parameters when implementing a GA, as this affects the results. There are many methods to find the best set of parameters. We tried many parameter sets at different ranges to tune the algorithm, and selected w = 0.1, σ = 30, and U = 110, based on the minimum returned fitness values. We temporarily set G to reach 500 generations. To determine how job similarity affects completion time, we consulted experienced engineers and defined:

$$reduct\,(z) = \begin{cases} 0.3 & \text{if } z \geq 0.75 \\ 0.2 & \text{if } 0.5 \leq \mathrm{z} < 0.75 \\ 0.1 & \text{if } 0.25 \leq z < 0.75 \\ 0 & otherwise \end{cases}$$

GA does not guarantee an optimum global result, as the results vary by the initialization. We sought the best result, so we ran the algorithm with the best set of parameters. Results of 15 executions with various initialization values are shown in Tab. 1. The fitness value is affected by three criteria, each with a different range of values. To avoid bias between different target values when calculating E and O distance, we normalized all data on dimensions that are not simple in the range [0, 1]. We could retrieve the same fitness value, but the objective functions returned different values. The execution time ranged from 1 to 1.5 min. Fig. 3 displays the fitness values $P_i^{(g)}.fitness$ and corresponding objective functions defined in Eqs. (1)–(3) as they changed over generations. We can observe that fitness values always decrease over generations because the selection always keeps the individual with the best fitness. The objective function's values can be incremented or decremented at a given generation when the fitness value between them does not change.

Table 1: Obtained values corresponding to 15 executions.

#Execution	Fitness	Critical path	Cost	Quality score	Time execution (ms)
1	0.469310668	565	304380	6506	72349.2
2	0.470994549	566	288005	6467	72950.64
3	0.467898598	557	291385	6462	68568.08
4	0.474917804	564	287875	6431	74604.88
5	0.471262737	563	296110	6469	80553.76
6	0.468132092	555	293610	6457	77132.16
7	0.474278761	562	307070	6464	79134.16
8	0.472727013	564	302435	6474	79376.64
9	0.473674097	567	303835	6481	76480.88
10	0.469917353	561	296525	6472	73631.6
11	0.473267236	571	293195	6479	98902.72
12	0.473042948	556	290755	6420	74950.96
13	0.470933679	561	297235	6466	83810.72
14	0.469091138	555	297310	6457	87889.76
15	0.472524921	561	306585	6472	84133.28

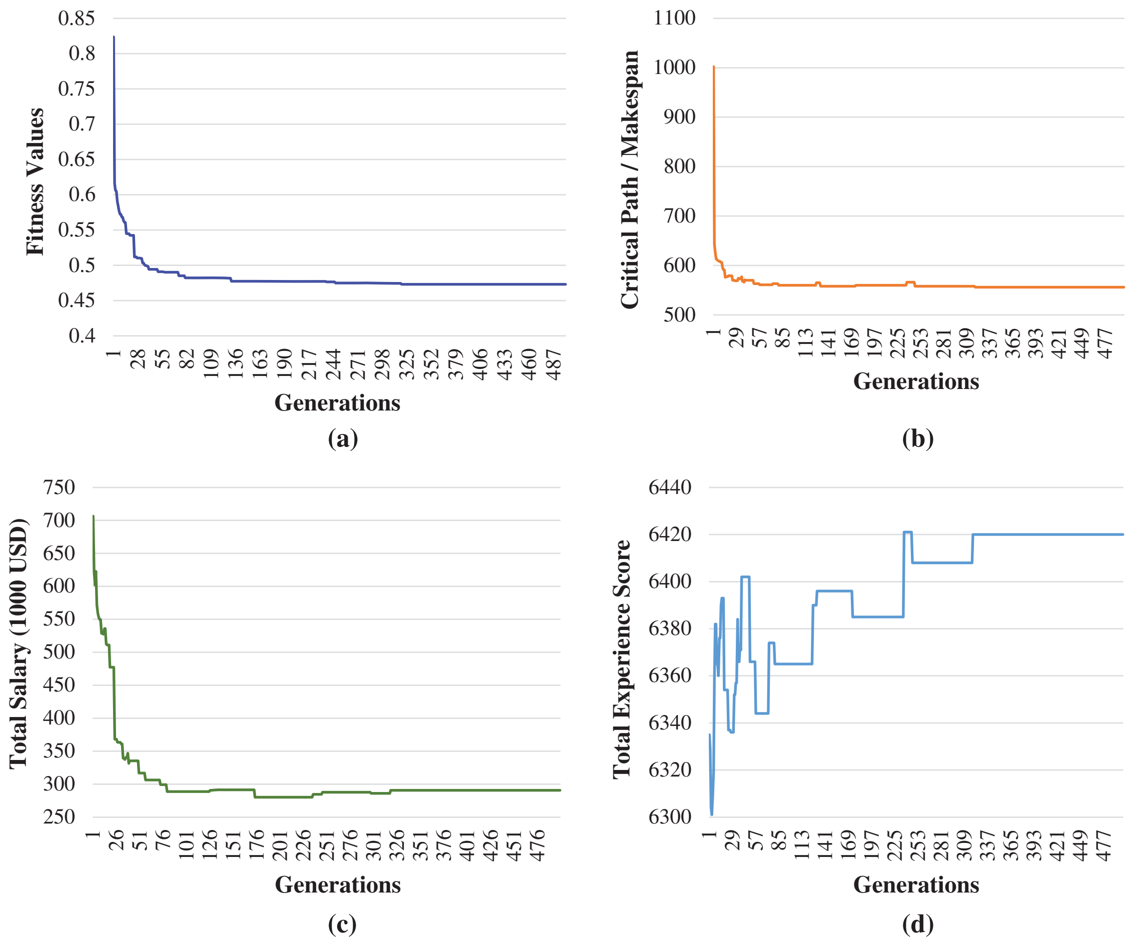


Figure 3: (a) Fitness values (b) Critical path objective values, (c) Total salary objective values, (d) Total experience score objective values; changing over generations

A schedule for each candidate for each time unit is shown in Fig. 4. A line represents a candidate, and columns represent the units of times. Assigned jobs have colors different from the background. We can notice that some members were used for a long time, while most were used occasionally. We checked with data on the availability of members. Members 3, 16, and 20 are those with the most registered time on the project, which explains why they took on much important work.

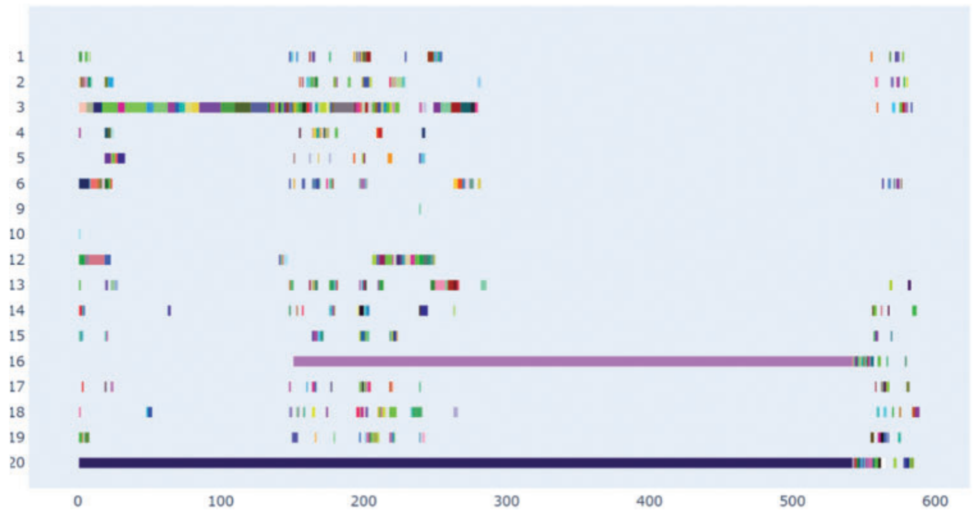


Figure 4: Generated schedule of the candidates. Vertical axis displays the workers and horizontal axis displays the time units

Tasks related to configuration and management of customer relationships and business processing support are lengthy. Fig. 5 shows the number of tasks received by each project member. Some members assigned zero or one tasks can be removed from the project candidates list, depending on conditions.

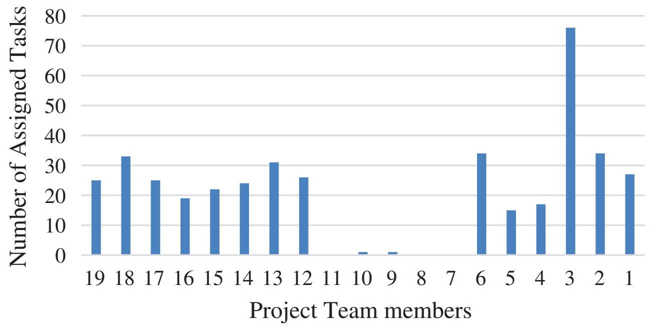


Figure 5: Number of assigned tasks to corresponding project team members

To test whether the proposed algorithm can find global solutions on smaller datasets, we created a smaller dataset with nine tasks, two candidates, and six skills from the standard dataset, then we compared the brute-force results with the proposed GA. The comparison results are shown in Tab. 2. The GA and the exhausting algorithm result return the same fitness values.

Table 2: Comparison result between the proposed algorithm and the brute force algorithm

Algorithm	Fitness value	Time execution (ms)
Genetic algorithm	0.826330532212885	659
Brute force	0.826330532212885	335787

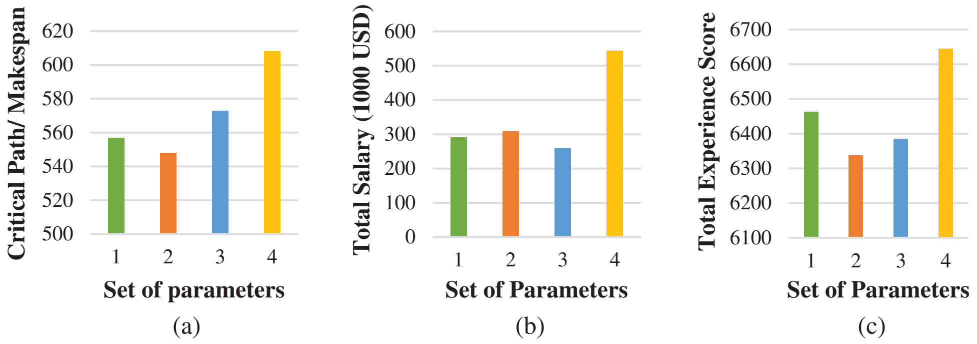


Figure 6: Execute algorithm with different parameters. (a) Shows the critical path/makespan values (***O*1**) corresponding to different sets of parameters. (b) Displays the total salary (***O*2**) corresponding to different sets of parameters. (c) Illustrates the sum of experience scores (***O*3**) of hired candidates on their assigned tasks. Green Chart $\boldsymbol{w_1 = 1, w_2 = 1, w_3 = 1}$; Orange Chart $\boldsymbol{w_1 = 20, w_2 = 1, w_3 = 1}$; Blue Chart $\boldsymbol{w_1 = 1, w_2 = 20, w_3 = 1}$; Yellow Chart $\boldsymbol{w_1 = 1, w_2 = 1, w_3 = 20}$

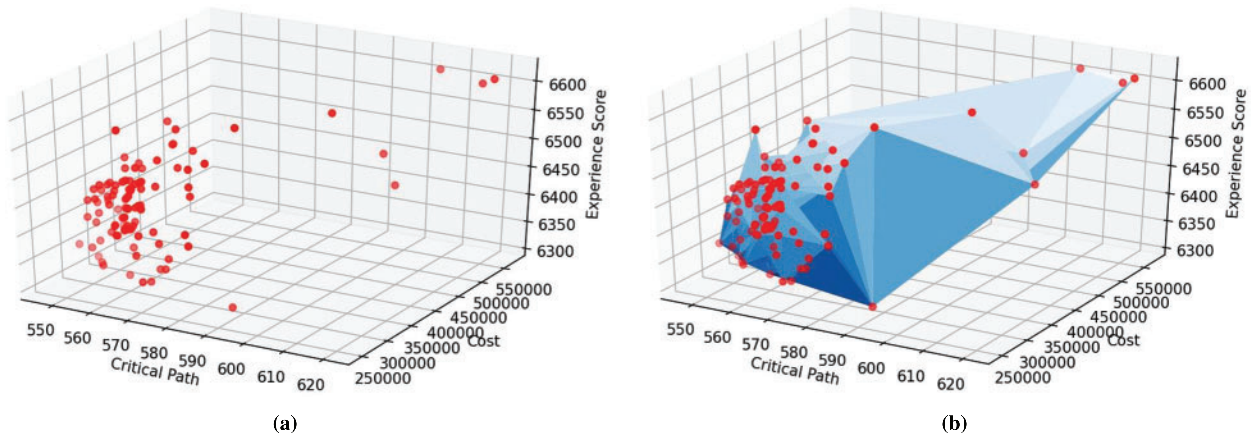


Figure 7: Collected solutions corresponding to 100 executions with different weighted parameters: (a) Solutions in the decision space; (b) Practical Pareto frontier obtained from 100 solutions in the decision space

Suppose decision-makers want to define preferences based on different objectives. They may change the values of weight parameters w_i to meet different expectations. Fig. 6 illustrates the returned results using different sets of parameters for various preferences. The objective is to minimize critical path (***O*1**), so the green bar shows the lowest value when the weight w_1 is at its maximum. The minimum value of total cost (***O*2**) is achieved with a greater value of w_2 (blue bar). The third target function maximizes full experience scores (***O***3), so a greater value of w_3 increases the score (orange bar).

To illustrate a solution space, we executed the algorithm 100 times with different weights of each objective function. It is hard to guarantee this frontier is Pareto optimal. However, many executions with different parameters provide a visualization of solutions offered by the tool in the decision space. We can consider it as the practical version of the boundary. Fig. 7 shows the Pareto frontier constructed from the practical result.

5 Conclusions

We proposed an approach to RCPSP. The model attempts to minimize critical paths and costs, and to maintain the quality of the product. The model is subject to classical RCPSP problem constraints, and some additional conditions in real software development projects. The model's goals are generally consistent with the project's planning. The model integrates the similarity of tasks and employees' available working plans, and therefore can be used generically for software projects, and not only applied to the specific cases cited.

Compared to other studies [5], the proposed approach brings two main benefits: (1) The model is formulated to allow decision-makers to understand and validate the structure. We combined both approaches to the proposed MOP, each fitting a different decision-making strategy. The weighted parameter-based method enables the decision-maker to customize the weights of target functions to flexibly achieve a final solution. Compromise programming is crucial in navigating the search, even when decision-makers cannot assign preferences to objective functions. (2) To solve the combinatorial optimization problem, we designed a new GA scheme that derives the individual fitness objective functions. Experimental results show that the introduced algorithm searches well for the solution by navigating the defined distance-based method.

The problem under consideration does not require finding solutions in spaces with large dimensions. However, optimal solutions may lie in high-dimensional space in other cases. It is challenging to visualize the Pareto frontier. In these situations, compromise programming is more suitable. We hope to refine the model to consider new goals and constraints in many situations. Recent methods of evolutionary algorithms show great promise in improving the quality of search results.

Acknowledgement: We thank LetPub (www.letpub.com) for its linguistic assistance during the preparation of this manuscript.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] P. Brucker, A. Drexl, R. Mohring, K. Neumann and E. Pesch, "Resource-constrained project scheduling: Notation, classification, models, and methods," *European Journal of Operational Research*, vol. 112, no. 1, pp. 3–41, 1999.

[2] C. Artigues, S. Demassey and E. Neron, *Resource Constrained Project Scheduling: Models, Algorithms, Extensions and Applications*. London, UK: Wiley-ISTE Ltd., pp. 1–288, 2008. https://onlinelibrary.wiley.com/doi/book/10.1002/9780470611227.

[3] C. Barnhart and G. Laporte, *Handbooks in Operations Research and Management Science: Transportation*, 1st ed., vol. 14. North Holland: Elsevier, pp. 1–784, 2007.

[4] S. Hartmann and D. Briskorn, "A survey of variants and extensions of the resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 207, no. 1, pp. 1–14, 2010.

[5] I. S. Ben and T. Yiliu, "A survey in the resource-constrained project and multi-project scheduling problems," *Journal of Project Management*, vol. 5, no. 2, pp. 117–138, 2020.

[6] F. Habibi, F. Barzinpour and S. J. Sadjadi, "Resource-constrained project scheduling problem: Review of past and recent developments," *Journal of Project Management*, vol. 3, no. 2, pp. 55–88, 2018.

[7] M. Vanhoucke and J. Coelho, "A tool to test and validate algorithms for the resource-constrained project scheduling problem," *Computers & Industrial Engineering*, vol. 118, no. 3, pp. 251–265, 2018.

[8] D. A. V. Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithms: analyzing the state-of-the-art," *Evolutionary Computation*, vol. 8, no. 2, pp. 125–147, 2000.

[9] V. V. Vazirani, Minimum makespan scheduling. In: *Approximation Algorithms*. Berlin, Heidelberg: Springer, pp. 79–83, 2003.

[10] N. Kumar and D. P. Vidyarthi, "A model for resource-constrained project scheduling using adaptive PSO," *Soft Computing*, vol. 20, no. 4, pp. 1565–1580, 2016.

[11] E. Arian, "A new approach for solving resource constrained project scheduling problems using differential evolution algorithm," *International Journal of Industrial Engineering Computations*, vol. 7, no. 2, pp. 205–216, 2020.

[12] A. Laurent, L. Deroussi, N. Grangeon and S. Norre, "A new extension of the RCPSP in a multi-site context: Mathematical model and metaheuristics," *Computers & Industrial Engineering*, vol. 112, no. 6, pp. 634–644, 2017.

[13] C. Stiti and O. B. Driss, "A new approach for the multi-site resource-constrained project scheduling problem," *Procedia Computer Science*, vol. 164, no. 1, pp. 478–484, 2019.

[14] S. C. Ali, "Resource tardiness weighted cost minimization in project scheduling," *Advances in Operations Research*, vol. 2017, pp. 1–11, 2017.

[15] D. H. Quoc, N. T. Loc, N. D. Cuong and X. Naixue, "Effective evolutionary algorithm for solving the real-resource-constrained scheduling problem," *Journal of Advanced Transportation*, vol. 2020, no. 2, pp. 1–11, 2020.

[16] A. H. Hosseinian and V. Baradaran, "Detecting communities of workforces for the multi-skill resource-constrained project scheduling problem: A dandelion solution approach," *Journal of Industrial and Systems Engineering*, vol. 12, pp. 72–99, 2019.

[17] B. A. Nadjafi, "Resource constrained project scheduling subject to due dates: Preemption permitted with penalty," *Advances in Operations Research*, vol. 2014, pp. 1–10, 2014.

[18] S. Ahmadpour and V. Ghezavati, "Modeling and solving multi-skilled resource-constrained project scheduling problem with calendars in fuzzy condition," *Journal of Industrial Engineering*, vol. 15, no. 1, pp. 179–197, 2019.

[19] C. Kellenbrink and S. Helber, "Scheduling resource-constrained projects with a flexible project structure," *European Journal of Operational Research*, vol. 246, no. 2, pp. 379–391, 2015.

[20] D. Joshi, M. L. Mittal, M. K. Sharma and M. Kumar, "An effective teaching-learning-based optimization algorithm for the multi-skill resource-constrained project scheduling problem," *Journal of Modelling in Management*, vol. 14, no. 4, pp. 1064–1087, 2019.

[21] A. Tesch, "A polyhedral study of event-based models for the resource-constrained project scheduling problem," *Journal of Scheduling*, vol. 23, no. 2, pp. 233–251, 2020.

[22] M. J. Anzanello and F. S. Fogliatto, "Learning curve models and applications: Literature review and research directions," *International Journal of Industrial Ergonomics*, vol. 41, no. 5, pp. 573–583, 2011.

[23] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, 5th ed., vol. 21. Berlin, Germany: Springer Publishing Company, 2006.

[24] S. Elsayed, R. Sarker, T. Ray and C. C. Coello, "Consolidated optimization algorithm for re-source-constrained project scheduling problems," *Information Sciences*, vol. 418, no. 1, pp. 346–362, 2017.

[25] S. Tao and Z. S. Dong, "Multi-mode resource-constrained project scheduling problem with alterna-tive project structures," *Computers and Industrial Engineering*, vol. 125, no. 6, pp. 333–347, 2018.

[26] H. Zhang, H. Xu and W. Peng, "A genetic algorithm for solving RCPSP," in *Int. Symp. on Computer Science and Computational Technology*, Shanghai, China, pp. 246–249, 2008.

[27] V. P. Van and M. Vanhoucke, "A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 201, no. 2, pp. 409–418, 2010.

[28] R. Carlos and R. Tahir, Chapter five compromise programming. In: *Developments in Agricultural Economics*. Vol. 11. Amsterdam, Netherlands: Elsevier, pp. 63–78, 2003.

[29] C. L. Hwang and A. S. M. Masud, *Multiple Objective Decision Making, Methods and Applications: A State-of-the-art Survey*, 1st ed., vol. 164. Berling Heidelberg: Springer-Verlag, pp. 1–358, 1979.

[30] M. Zeleny, *Compromise Programming, Multiple Criteria Decision Making*, J. L. Cochrane, M. Zeleny (Eds.), Columbia, South Carolina, United States: University of South Carolina Press, pp. 262–301, 1973.

[31] T. S. Ngo, N. A. Bui, T. T. Tran, P. C. Le, D. C. Bui *et al.,* "Some algorithms to solve a bi-objectives problem for team selection," *Applied Sciences*, vol. 10, no. 8, pp. 1–19, 2020.

[32] H. Xiong, Q. Shi, X. Tao and W. Wang, "A compromise programming model for highway maintenance resources allocation problem," *Mathematical Problems in Engineering*, vol. 2012, pp. 1–11, 2012.

[33] B. Poff, A. Tecle, D. G. Neary and B. Geils, "Compromise programming in forest management," *Journal of the Arizona-Nevada Academy of Science*, vol. 42, no. 1, pp. 44–60, 2010.

[34] N. T. Son, J. Jaafar, I. A. Aziz and B. N. Anh, "A compromise programming for multi-Objective task assignment problem," *Computers*, vol. 10, no. 2, pp. 1–15, 2021.

[35] N. T. Son, J. Jaafar, I. A. Aziz and B. N. Anh, "Meta-heuristic algorithms for learning path recommender at MOOC," *IEEE Access*, vol. 9, pp. 59093–59107, 2021.

[36] C. A. Coello Coello, G. B. Lamont and D. A. Veldhuizen, *Evolutionary Algorithms for Solving Multi-objective Problems*, 2nd ed., vol. 5. US: Springer, pp. 79–104, 2007.

[37] K. F. Man, K. S. Tang and S. Kwong, "Genetic algorithms: Concepts and applications," *IEEE Transactions on Industrial Electronics*, vol. 43, no. 5, pp. 519–534, 1996.