Received March 16, 2021, accepted April 3, 2021, date of publication April 9, 2021, date of current version April 23, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3072222

Meta-Heuristic Algorithms for Learning Path Recommender at MOOC

NGO TUNG SON[1,2], JAFREEZAL JAAFAR[1], (Senior Member, IEEE), IZZATDIN ABDUL AZIZ[1], (Member, IEEE), AND BUI NGOC ANH[2]

[1]Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Seri Iskandar 32610, Malaysia
[2]Information and Communication Technology Department, FPT University, Hanoi 100000, Vietnam

Corresponding author: Ngo Tung Son (sonnt69@fe.edu.vn)

This work was supported by the FPT University, Hanoi, Vietnam under Grant 1097/QD-DHFPT.

ABSTRACT Online learning platforms, such as Coursera, Edx, Udemy, etc., offer thousands of courses with different content. These courses are often of discrete content. It leads the learner not to find a learning path in a vast volume of courses and contents, especially when they have no experience in advance. Streamlining the order of courses to create a well-defined learning path can help e-learners achieve their learning goals effectively and systematically. The learners usually ask the necessary skills that they expect to earn (query). The need is to develop a recommender system that can search for suitable learning paths. This study proposes a multi-objective optimization model as a knowledge-based recommender. Our model can generate an appropriate learning path for learners based on their background and job goals. The recommended studying path satisfies several learner criteria, such as the critical learning path, number of enrollments, learning duration, popularity, rating of previous learners, and cost. We have developed Metaheuristic algorithms includes the Genetic Algorithm (GA) and Ant Colony Optimization Algorithm (ACO), to solve the proposed model. Finally, we tested proposed methods with a dataset consisting of Coursera's courses and Vietnam work's jobs. The test results show the effectiveness of the proposed method.

INDEX TERMS Learning path, Knowledge-based recommendation, Knowledge graph, multi-objective optimization, compromise programming, genetic algorithm, ant colony optimization algorithm.

I. INTRODUCTION

A. RESEARCH CONTEXT

Currently, the speed of knowledge expansion is breakneck. Therefore, learning styles also need to be adapted to increase efficiency. Online learning and Massive Open Online Courses (MOOCs) are revolutionizing education [1]. Many companies offered online studying platforms, such as Coursera, Udemy, EdX, Udacity. They allow the course providers to publish their online courses in many areas. It opens up unprecedented learning opportunities for learners. However, it is difficult for learners, especially those who are difficult to get experts advice, may not find a suitable learning path among many different courses and content. Therefore, the task is to advise learners in an appropriate direction based on their aspirations about skills, nevertheless satisfying learners' available constraints. Selecting the right courses to compose a suitable learning path is a complicated task. It requires considering the relationship between the courses and the student preferences in recommendation model development.

The recommender takes the required skills and knowledge (learning outcomes or objectives) of the job(s) and existed skills/knowledge of the learner as the input. It then searches for the learning path, which is the set of courses and their order in the path. The recommended learning path contains at least learning outcomes matched to the job query as the output. The learners need evaluations on their current proficiency by experts before the pathway counseling process. After both, the starting point (current level) and the target (expected level) have been determined, the system can create a learning path that meets the student's expectations, as shown in **Figure 1**. For example, to work as a java web developer, learners need to know "object-oriented programming," "java programming language," "Web design," "java web framework." A roadmap consisting of dependent courses like

The associate editor coordinating the review of this manuscript and approving it for publication was Xujie Li.

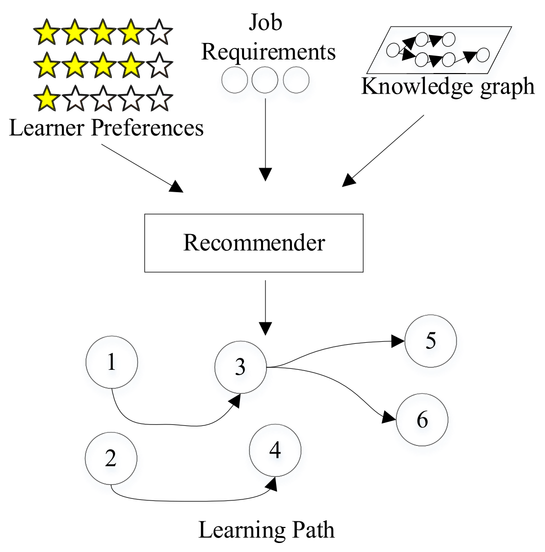

FIGURE 1. **Learning path recommendation system.**

"OOP with Java," "Java Web application development" can be consulted, and independent courses like "front-end web development."

The learning path search problem involves the courses and their dependencies, learner preferences, and recommendation criteria to indicate the solutions. These signals tell the recommender is in the forms of Knowledge-based Recommender Systems [2]. It is not easy or suitable to apply traditional recommender techniques such as content-based filtering or collaborative filtering [3]. Fortunately, we can explicitly identify the good recommendation for the learning path due to a clear understanding of affected criteria to the learner's learning conditions. This research aims to build a recommender engine as an optimization solver that can generate a suitable learner route.

B. PREVIOUS RESEARCHES

Many researchers are working to build a learning path consulting system for online learning. They came up with many different strategies, but they can be categorized into two groups: (1) data mining-based techniques and (2) knowledge graphs. Based on data mining, the researchers often seek to detect similarity points between factors in the system, such as learner or material, to make recommendations. These recommendations can be a learning path or a solution to improve learning efficiency. Wong and Looi [4] introduced an ant-colony algorithm to find a learning path for the learner. They compute the similarity levels between individual alumni and the current learner to make recommendations for the target learner. Hsieh and Wang [5] introduced an e-learning System that can create a relationship hierarchy of learning materials to create a learning path using data mining. Jugo *et al.* [6]. developed an Intelligent Tutoring System with four core modules for learning path recommendation. It allows users to interact with the system without using Data Mining interfaces. Tam *et al.* [7]. introduced a learning path recommendation system based on concept clustering and heuristic-based search algorithm. Chen [8] constructs a personalized e-learning system that can generate appropriate learning paths by mining the individual learner's pretest and learning performance data. Tang and McCalla [9]. used the Nearest-Neighbor search to recommend the papers for users. Hsu [10] uses content-based filtering, collaborative filtering, and data mining techniques to analyze the students' reading data for recommending the reading lessons. Many studies use data mining, ignoring the relationships between the courses, people and learning paths, and other learners' interested criteria. This leads the generated recommendations are often redundant or sometimes unrelated to the needs of the learners.

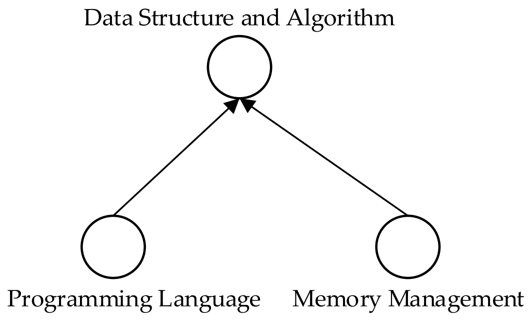

FIGURE 2. **Example of knowledge graph.**

Besides the data mining–based approach, another approach uses knowledge graphs for applying these dependencies to the learning path recommendation model. The knowledge graph can be formed by the learning outcomes from the curriculum as shown in **Figure 2.** This method pays more attention to the dependence on learners' level/perception. It can come up with solutions closer to learners. The learning path consulting process is like a decision-making process when the learners are decision-makers with their concerns. Kurilovas *et al.* [11]. recommend the learning path by the sequences of learning objects according to learners' preferences including their prerequisite knowledge level, learning path, learning object, and learning style. The proposed method is sufficient to search for the learning paths. However, their research does not treat the recommender as a decision-making process with many considerations from the decision-makers, here the learners. Wan and Niu [12]. created a recommender for learning paths based on the graph where its vertex is knowledge units. Shmelev *et al.* [13]. represent the learning path as the learning outcomes sequence, using a genetic algorithm to generate the paths. Zhu *et al.* [14]. introduced a multi-objective optimization (MOP) for learning path recommendations where they have used linear Scalarizing to approach the MOP. They take into account many goals to construct the recommended learning path. However, these objectives are not to fully cover the different aspects surrounding MOOCs such as cost, rating of community, and popularity in recent. Shi *et al.* [15]. developed a recommendation model based on a multidimensional knowledge graph where nodes in the same dimension represented by inner-class relationships. Intra-class relationships connect these dimensions. In comparison to previous studies, the authors added a hierarchical structure in the interdependence of learning outcomes. However, their search problem did not involve the relationships between learning outcomes and courses, which will expand the search space.

The use of knowledge graphs allows thorough matching of knowledge required with the learning path. However, the proposed models are built on aspects that are not in line with the MOOC Courses advisory context. Learners who study with MOOC have several concerns, such as critical path, learning effort, and cost. The advisory systems merely indicate the skills to be acquired, but the guide to approaching these skills not included in the recommendation package. Previous learner assessments also played an important role in assigning the course to the learning path. Each learner has different learning needs. Hence providing a useful decision-making tool that allows them to consider various options when finding a suitable learning path in both cases that learners may/may not indicate their preferences is also not adequately considered. Based on summarizing the previous study, we construct a multi-objective model as a knowledge graph-based recommender model for MOOC's learning path.

The learning path recommendation is classified as both combinatorial optimization and NP-Hard problem [16]. It is not easy to use an exact algorithm to solve the problem, but metaheuristic algorithms [17]. For example, Zhu *et al.* [14]. designed a Genetic Algorithm to solve multi-criteria optimization for proposed recommender. Zhao *et al.* [18]. developed an Ant Colony Optimization Algorithm for Recommendation of Micro-Learning Path. Due to the different objectives and constraints of the proposed model. It is no possibility of applying the existing algorithms to our problem directly. Therefore, we design the new schemes of Meta-heuristic to construct the recommender. We also implement algorithms to perform parallel computations on multiple processing cores.

C. CONTRIBUTIONS

In this study, we present an approach to construct a learning path recommendation system at MOOC. This recommender is essentially a multi-objective optimization model. The model works based on the knowledge graph to show the linkages between learning outcomes and course dependencies in model designing. We use a combination of linear scalarizing approach and compromise programming for the proposed MOP. It allows the consultation process to become a decision-making process. The learner is a decision-maker and can express his or her preferences on different goals by defining the weights to determine the objectives' impacts via the distance function. The objective functions (including learning critical path, learning effort, average rating by the community, the recent number of enrollments, cost) are built around the purpose of recommending appropriate MOOC courses for learners according to different learning scenarios, instead of drawing the graph of needed learning outcomes. We designed a Genetic Algorithm and Ant Colony Optimization algorithm to solve the proposed model. We use data collected from Coursera to evaluate our algorithm.

This study benefits researchers and engineers to develop a better advisory system for studying at MOOC. When the research results are widely deployed, it is possible to improve learners' learning abilities and opportunities. Play a role in promoting the quality of human resources in society. The rest of this paper is organized as follows. The proposed model and algorithm are respectively described in Sections 2 and 3. To evaluate the proposed approach, we display the experiments and discussion in Section 4. Finally, section 5 offers a conclusion.

II. PROPOSED RECOMMENDATION MODEL

A. MULTI-OBJECTIVE OPTIMIZATION MODEL

To determine the goals, learners need to specify the jobs they want to achieve. Each job has been tagged with required learning outcomes—each course matched with learning outcomes that it provides to learners and other decision criteria when completing. The recommendation model can be formulated as:

We denote C as the number of available courses.

S is the number of available learning outcomes.

P represents the set of predefined jobs (user can choose one or more jobs).

B is the budget of the user.

$R = \{r_{p,s} | r_{p,s} \{\in\ 0, 1\}, p = 1 \ldots P, s = 1 \ldots S\}$ is a matrix to represent the required learning outcomes of the jobs, where $r_{p,s} = 1$ means the learning outcome s^{th} required by the job p^{th} and $r_{p,s} = 0$ otherwise.

$O = \{o_p | o_p \in \{0, 1\}, p = 1 \ldots P\}$ is a vector that represents the registered jobs of the user, where $o_p = 1$ if the learner register for job p^{th} and $o_p = 0$ otherwise.

$V = \{v_s | v_s \in \{0, 1\}, s = 1 \ldots S\}$ defines a vector that illustrate the learning outcomes that the user achieved before the learning process, where $v_s = 1$ means the user already gained learning outcome s^{th} and $v_s = 0$ otherwise.

$L = \{l_{c,s} | l_{c,s} \in \{0, 1\}, c = 1 \ldots C, s = 1 \ldots S\}$ describes the adjacency matrix, where $l_{c,s} = 1$ determines that course c^{th} provide learning outcome s^{th} and $l_{c,s} = 0$ otherwise.

$H = \{h_c | h_c \in \mathbb{R}^+, c = 1 \ldots C\}$ where h_c denotes the averaging rating of the community on the course c^{th}.

$Q = \{q_c | q_c \in \mathbb{N}^+, c = 1 \ldots C\}$ where q_c is the number of recent number of enrollment of course c^{th}.

$P = \{p_c | \in \mathbb{R}^+, c = 1 \ldots C\}$ where p_c is price of course c^{th}.

$Z = \{z_{i,j} | z_{i,j} = \{0, 1\}, , i = 1 \ldots C, j = 1 \ldots C\}$ is adjacency matrix to present the knowledge graph, where $z_{i,j} = 1$ indicates that learning outcome i^{th} must be archived before learning outcome j^{th} and $z_{i,j} = 0$ means they have no dependency.

$K = \{k_c | k_c \in \mathbb{N}^+, c = 1 \ldots C\}$ is the advertised duration to complete course c^{th}.

Denote X as the decision variables to represent a adjacency matrix of a directed graph for the recommended learning path. Where $X = \left\{x_{i,j} \,|\, x_{i,j} \in \{0,1\}, i = 1 \ldots C, j = 1 \ldots C\right\}$ with $x_{i,j} = 1$ shows that course i^{th} starts before course j^{th} and $x_{i,j} = 0$ means they have no dependency.

D is the set of vertexes of the graph represented by X. In other words, D presents the set of course that recommend to user. $D = \{d_c | d_c \in \{0,1\}, c = 1 \ldots C\}$ where $d_c = 1$ if course c^{th} is a vertex of graph represented by X, $d_c = 0$ otherwise.

$T = \{t_c^{complete} | t_c^{complete} \in \mathbb{N}^+, c = 1 \ldots C, \}$ presents the actual time user complete the courses with $t_c^{complete}$ denotes the time when the learner complete course c^{th}.

$$t_c^{complete} = \begin{cases} -1 & if\ d_c = 0 \\ t_c^{complete} \geq 0 & otherwise \end{cases}$$

The objective functions declared as follows:

- Minimizing the critical learning path for the learners who need to complete the learning path in the shortest time. This is targeted to the expectation of learners who need the achievements urgently.

minimize
$$\left(\begin{aligned} &f_1(X) \\ &= \max_{c=1\ldots c} \left(\left(1 - \min\left(1, \sum_{d=1}^{C} x_{c,d}\right) * d_c \right) * t_c^{complete} \right) \end{aligned} \right)$$

- Minimize the learning time. This aim to save total the effort of studying.

$$\text{minimize}(f_2(X) = \sum_{c=1}^{C} d_c * k_c)$$

- Course with higher ratings have a higher chance of being recommended. Community assessment is one of the essential criteria to determine course quality.

$$\text{maximize}\left(f_3(X) = \frac{\sum_{c=1}^{C} d_c * h_c}{\sum_{c=1}^{C} d_c} \right)$$

- Courses with a higher number of recent enrollments have a higher chance of being recommended. The course has many students demonstrating it as a quality course. However, to avoid the domination of courses that appear first in the search process. We only consider the number of enrolments within a defined period.

$$\text{maximize}\left(f_4(X) = \frac{\sum_{c=1}^{C} d_c * q_c}{\sum_{c=1}^{C} d_c} \right)$$

- The learning path consists of lower-cost courses that are gotten a higher chance of being recommended.

$$\text{minimize}\left(f_5(X) = \sum_{c=1}^{C} d_c * p_c \right)$$

The recommended learning path must satisfy the below constraints:

- Courses must be organized according to the dependencies of the learning outcomes defined in the knowledge graph. One cannot enter the course contains entry requirements that he/she does not archive in the completed courses.

$$a_i^c \geq e_i^c \quad \forall i = 1 \ldots S, c = 1 \ldots C \qquad (CO_1)$$

where:

$$\begin{aligned} a^c &= \{a_i^c | a_i^c \in \{0,1\}, i = 1 \ldots S\} \\ e^c &= \{e_i^c | e_i^c \in \{0,1\}, i = 1 \ldots S\} \end{aligned}$$

l_i is vector row i^{th} of the matrix L.
z_i is vector column i^{th} of matrix Z.
x_i is vector column i^{th} of matrix X.

$$\begin{aligned} e^c &= \underset{i=1\ldots s}{\text{combine}} (\vec{l_c}, \vec{z_i}) \\ a^c &= \text{combine}(\underset{d=1\ldots c, d \neq c}{\text{combine}} (\vec{l_d}), V) \end{aligned}$$

where $C = \text{combine}(A, B)$ iff $|A| = |B| = |C|$ and $C_i = \max(A_i, B_i) \forall i = 1 \ldots |C|$

- The qualifications of the registered job must be met. After the recommended courses are completed, all of the learning outcomes to satisfy the query must be reached.

$$r_{p,s} \leq \max\left(\max_{c=1\ldots c} (d_c * l_{c,s}), v_s \right) \qquad \forall p = 1..P, s = 1 \ldots S \qquad (CO_2)$$

- The total cost for the courses should not exceed the budget.

$$\sum_{c=1}^{C} d_c * p_c \leq B \qquad (CO_3)$$

B. APPROACHES TO MOP

There are two major approaches to the multi-objective problem [19], no-preference, and preference. One says that there is never one best approach to all types of multi-objective mathematical programming problems [20]. In this section, we present the approach we use to take the problem of single-objective optimization problem.

1) COMPROMISE PROGRAMMING

Compromise programming is a well-known non- preference approach for MOP. It works based on an assumption that no decision-maker exists. Instead of asking the decision-maker to assign a priority to each goal, an expectation point is chosen [20]. The mission of the model now is to find the solutions closest to this expected point. We have been success used compromise programming for our MOP in the team selection problem [21] and task assignment problem [22]. For the considered problem compromise programming is described as follows.

$E = \{E_i | i = 1..5\}$ denotes the expected point. Where:

$$E_i = 0 \quad \forall i = 1, 2, 5$$

$$E_3 = \sum_{c=1}^{C} h_c$$
$$E_4 = \sum_{c=1}^{C} q_c$$

$O = \{O_i | i = 1..5\}$ denotes the actual solution. Where:

$$O_i = f_i(X) \quad \forall i = 1 \ldots 5$$

The objective functions can be rewritten as:

$$\text{minimize}\,(distance\,(E, O)) = \sqrt{\sum_{i=1}^{5} (E_i - O_i)^2}$$

2) LINEAR SCALARIZING

Compromise programming can be seen as an equally important goal. However, in this situation, the learners completely determine the preference for each of their goals. So we use linear scalarizing a preference-based method to transform the problem on a single-objective problem, which is similar to *Zhu et al.* [14]. The objective functions can be reformulated as follows:

$$\text{minimize}\left(\sum_{i=1}^{5} (w_i * f_i(X))\right)$$

where $w_i \in \mathbb{R}^+$ is the weight parameter of the objective i^{th}.

To reuse the good characteristic of both methods. The weight parameters are added to the dimension of decision space in compromise programming instead of parameterizing the objective function [23]. The distance function's size is influenced by spatial dimensions, to be manipulated by the weights of each dimension:

$$minimize\,(obj = (distance\,(E, O)))$$
$$= \sqrt{\sum_{i=1}^{5} w_i * norm\,(E_i - O_i)^2}$$

$\boldsymbol{w_i} \in \mathbb{R}^+$ now is the weighted parameter of the dimension $\boldsymbol{i^{th}}$ in the distance function. ***norm*** denotes the normalization function that used to rearrange the values of the dimensions to the same range.

III. PROPOSED ALGORITHMS

A. GENETIC ALGORITHM

Evolutionary algorithms (EA) are widely used as stochastic methods [24]. The genetic algorithm (GA) is one of the most popular in the EA family due to its speed and efficiency. The algorithm mimics natural selection, in which the fittest individuals are chosen for reproduction to produce the next generation's offspring. Here an individual represents a potential solution to the problem. The flow of the proposal is shown in **Figure 3**.

The $f_1(X)$ aims to minimize the learning critical path. This is the form of the makespan problem [25]. There are many solutions to this problem, but the simplest one is to consider learning path X as a graph. The two fakes nodes added are Head and Tail (the last node) with a duration equal to 0. The problem transformed to single start and ended problem. Finally, the critical path/makespan is computed by any traverse method such as Breadth First Search (BFS).

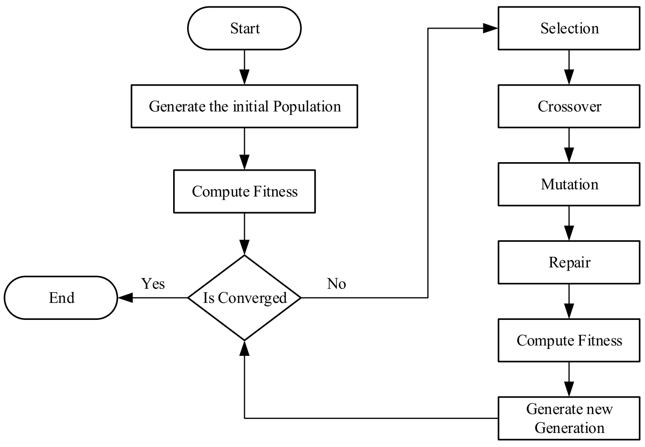

FIGURE 3. The flow of proposed Genetic algorithm.

The genetic algorithm's scheme can be described as follows:

- β denotes the number of generations to stop the algorithm.
- £ is the population size (number of individuals in the same population).
- δ stands as the mutation rate.
- $\theta^{(g)} = \left\{\theta_1^{(g)}, \theta_2^{(g)}, \ldots, \theta_{£}^{(g)}\right\}$ denotes the population with £ individuals at g^{th} generation.
  Where $\theta_i^{(g)} = \{\theta_{i,1}^{(g)}, \theta_{i,2}^{(g)}, \ldots, \theta_{i,C}^{(g)}\}$ is an individual of the population at g^{th} generation. The element $\theta_{i,index}^{(g)} = \{0, 1, \ldots, C\}\,\forall index = 1 \ldots C$.
  The *index* represents the order of course that appeared in the learning path. The course *index*-th is only considered to learn after courses that appear in front of it in $\theta_i^{(g)}$, only if it contains any dependent learning outcomes to these courses. 0 indicates that there is no course delivered at the slot. If the learning path contains ϑ courses, we have:
  $$\theta_{i,index}^{(g)} = \begin{cases} i \in \{1 \ldots C\} & if\ index \le \vartheta \\ 0 & otherwise. \end{cases}$$
- l_s is the row s^{th} of L.
- λ is selection rate.s
- η is the exchange rate of gens between $\theta_i^{(g)}$ and $\theta_j^{(g)}$
- r_s is the column s^{th} of R
- $is\ Respect\ (i) = \begin{cases} 1\ if\ CO_i isvalid \\ 0\ otherwise \end{cases} \quad \forall i = 1 \ldots 3$
- $dump_i = 1\ \forall i = 1 \ldots 5$
- $\theta_i^{(g)}.fitness = \sqrt{\sum_{i=1}^{5} w_i * \left(\frac{E_i - O_i}{dump_i}\right)^2}$
- $g = 1$ is the current generation.
- w is rate of elites of the population.

The algorithm contains 6 steps:

1. Initialize the first population

1.1. Randomly generate $\theta_i^{(1)} \forall i = 1 \ldots £$

$$\theta_{i,j}^{(1)} = rand\left(l_{s,j}, v_s, r_s\right) \quad \forall j = 1 \ldots C$$

where: r and $\left(l_{s,j}, v_s, r_s\right)$ randomly returns a course.

$$rand\left(l_{s,j}, v_s, r_s\right) = \begin{cases} j \in (1 \ldots C) \text{ randomly.} \\ \text{if } v_s = 0 \wedge \left(\sum_{p=1}^{P} r_{s,p} = 1\right) \\ \wedge\ is\ Respect\ (1) = 1 \\ 0 \quad otherwise \end{cases}$$

2. Selection.
   2.1 keep $\lambda * £$ individual that return the best fitness for next-generation.
   2.2 $b^g = \min_{i=1\ldots£} D_i^{(g)}.fitness$
   2.3 $dump_i = E_i - O_i$ if $g = 1 \forall i = 1 \ldots 5$
3. Crossover. Create $£ * (1 - \lambda)$ remaining individuals for the next generation. The following steps repeated until population at generation $(g + 1)^{th}$ fully constructed.
   3.1 Rearrange $\theta^{(g)}$ in descending $\theta_i^{(g)}.fitness$ order.
   3.2 Declare $Elite$ as the set contains top $£ * w$ items of $\theta^{(g)}$. Randomly choose $\theta_{father}^{(g)}, \theta_{mother}^{(g)}$ from $Elite$.
   3.3 Update $\theta_{father}^{(g)}, \theta_{mother}^{(g)}$ by a rate, denoted as ran. Where: $ran = rand([0, 1])$ return a random probability.
      3.3.1 If $(ran \leq \eta)$ Randomly exchange a gen with value 1 of $\theta_{father}^{(g)}$ and $\theta_{mother}^{(g)}$ to each other to create two new individuals for the next generation.
      3.3.2 If $(\eta < ran \leq \delta)$ perform mutation on $\theta_{father}^{(g)}$ or $\theta_{mother}^{(g)}$ (Described in step 5) to create two new individuals for the next generation.
      3.3.3 Otherwise consider $\theta_{father}^{(g)}$ and $\theta_{mother}^{(g)}$ as two new individuals for the next generation.
      3.3.4 Repair new created individuals. The Repair process described in step 4.
4. Repair: this phase take an individual ξ as the input. It contains 3 stages:
   4.1 Rearrange: ξ to $isValid(1) = 1$.
   4.2 Fix: generates new genes of ξ to $\sum_{i=1}^{3} isValid(i) = 3$. If there is not possible to fix ξ (no solution) then $\xi = 0$. Remove ξ from its population.
   4.3 Refine: Remove redundancies of the individual ξ.
5. Mutation: takes an individual ξ as input.
   5.1 Randomly exchange a selected course in ξ with its most similar course based on learning outcome they provide.
   5.2 Repair ξ.
6. Repeat 2, 3, and 4,5 and set $g = g + 1$ until $b^g = b^{g-1} = \ldots = b^{g-\beta}$.

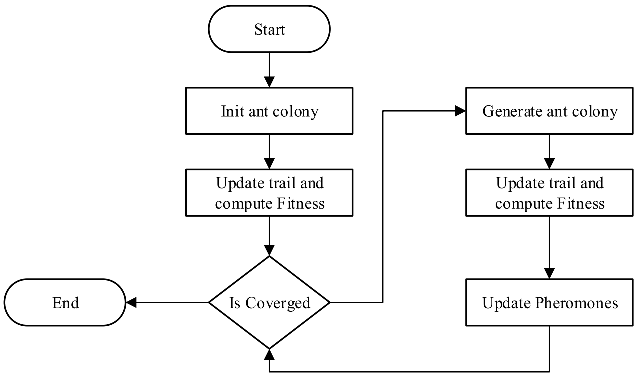

FIGURE 4. **The flow of proposed Ant Colony Optimization algorithm.**

B. ANT COLONY OPTIMIZATION ALGORITHM

The ant colony optimization algorithm is a technique to solve optimization problems. Using the multi-agent population (artificial ants) can find the right paths on the graphs. Here, the ants are inspired by the behavior of real ants. They communicate with each other using the pheromone [26]. The basic flow of the ACO is illustrated in **Figure 4**.

1. Initial LOs goal:
   1.1 Let $G = \{g_s \in \{0, 1\}, s = 1 \ldots S\}$ is a matrix to present the learning outcomes user need to archive
   1.2 Update $g_s = 1$ with the learning outcomes $r_{p,s} = 1$ that the registered jobs of user $o_p = 1$ required
   1.3 Update $g_i = 1$ with the learning outcomes $\{i\}$ must be archived before learning outcome $g_j = 1$ in knowledge graph $z_{i,j} = 1$
   1.4 Repeat 1.3 until every learning outcomes $g_s = 1$ required are in G
   1.5 Remove learning outcomes user archived before learning $g_s = \begin{cases} 1 \ if \ v_s = 0 \ and \ g_s = 1 \\ 0 \ otherwise \end{cases}$
2. Generate the ant colony and initial cost matrix and pheromones matrix:
   2.1 Let $A = \{a_{m,c} \in \mathbb{N}, m = 1 \ldots M, c = 1 \ldots C\}$ is a matrix of learned course each ant, where $a_{m,c} \geq 1$ mean course c^{th} is the $a_{m,c}^{th}$ course that m^{th} ant learned and $a_{m,c} = 0$ otherwise
   2.2 Let $Fitness = \{fitness_m \in \mathbb{R}^+, m = 1 \ldots M\}$ is a matrix to present fitness values of each ant
   2.3 Let $Cost = \{cost_c = \sqrt{\sum_{i=1}^{5} (E_i - f_i(C))^2}, i = 1 \ldots 5, c = 1 \ldots C\}$ is a matrix to present cost of an ant to learn c^{th} course
   2.4 Let $Ph = \left\{ph_{i,j} \in \mathbb{R}^+, i = 1 \ldots C, j = 1 \ldots C\right\}$ is a matrix to present density of pheromone on trial $i \to j \ if \ i \neq j \ and \ nest \to j \ if \ i = j$, default will be $ph_{i,j} = 0.1$
3. Update trail and compute fitness for ant a_m:
   3.1 Let $GX = G$ is a matrix to present the learning outcomes the ant need to archieve left
   3.2 Let $LOH = \{loh_s \in \{0, 1\}, s = 1 \ldots S\}$ is a matrix to present the learning outcomes ant currently have

3.3 Let $CR = \{cr_c \in \{0, 1\}, c = 1 \ldots C\}$ is an adjacency matrix of recommend courses to ant's current course c'^{th}, with $cr_c = 1$ if LOH satified learning outcome of course c^{th}

3.4 Calculate probability of each course

$$cr_c = 1 : prob_c = \frac{ph^{\alpha}_{c',c} * \frac{1}{cost_c}^{\beta}}{\sum_{cr_i=1} (ph^{\alpha}_{c',i} * \frac{1}{cost_i}^{\beta})}$$

3.5 Random $rd \in [0, 1)$ to choose course $cr_c = 1$ with cumulative distribution and update $gx_j = 0, loh_j = 1, a_{m,c} = 1$ if course c^{th} provide learning outcome j^{th}

3.6 Repeat 3.3 until $gx_s = 0 \forall s = 1 \ldots S$

3.7 Remove unnecessary courses from trail a_m

3.8 Compute fitness of trail

$$a_m : fitness_m = \sqrt{\sum_{i=1}^{5} w_i * \left(\frac{E_i - O_i}{dump_i}\right)^2}$$

4. Compute dump value with ant m^{th}, $fitness_m = \min(Fitness)$: $dump_i = E_i - f(a_m)_i \forall i = 1 \ldots 5$
5. Repeat these step until fitness converge:

5.1 Recreate matrix A and repeat 3

5.2 Decrease pheromones on every trail $ph_{i,j} = (1 - \rho) * ph_{i,j} \forall i, j = 1 \ldots C$

5.3 Let $INC = \left\{inc_m = \frac{Q}{fitness_m}, m = 1 \ldots M\right\}$ is a matrix to present the increased amount of pheromone by m^{th} ant

5.4 Increase pheromones on trails $i \rightarrow j$ for m^{th} ant $ph_{i,j} = ph_{i,j} + inc_m \forall m = 1 \ldots M, \forall i, j = 1 \ldots C$ if $a_{m,i} + 1 = a_{m,j} \& i, j > 0$.

C. PARALLEL SETTING FOR PROPOSED ALGORITHMS

Parallel processing is the most effective choice for tackling costly computations in MOP, especially with the cost reduction of high-speed multi-core processors [27]. To set the parallel running for both algorithms. We re-implement the computation of the particular agents that run on independent processing cores. The computations of each agent are considered tasks. Tasks are stored in a queue. Processes de-queue elements to process until the queue is empty. Most agents are doing the heaviest computation. Some other tasks need to be performed sequentially because it has to wait for all agents to finish before synthesizing information. The expected computation speed is a linear decrease in proportion to the number of cores used. The parallel setting for GA can be implemented by using different cores for each individual's genes selection process. The parallel setting for ACO is implemented by adding the asynchronous context to the exploration of the individual ant. The detail of these parallel schemes shown in **Appendix A** and **B**.

IV. EXPERIMENTS

A. EXPERIMENTAL DESIGN

To conduct experiments to evaluate the proposed model and algorithm. We crawled data of more than 169 courses from Coursera. We can gather the average rating for the dataset by a web crawler. The older courses that have been published for a long time can dominate the latest with a large number of enrollments. We re-crawl the data after 30 days to count the number of registrations over a recent period. Identifying about 342 learning outcomes and the relationship between them is arduous work. The reason is we didn't have an efficient mining tool to collect these data automatically. Therefore, all of the learning outcomes are determined and adjusted manually from the syllabus, course description, and course tags.

The topics of collected courses revolve around the field of software engineering. We collect 18 job calls related to software engineering, such as business analysis, back-end Java engineer, project manager, and the most famous recruitment portal in Vietnam. The Job descriptions list the required skills. However, we also need to match these requirements with the learning outcomes archived manually. The entire e-learner is assumed to be without any initial knowledge. We use this data in the parameters selection section and leave the weights for the objective functions equal. To test the model with different weights of the criteria according to decision-maker preferences, we clone 12 courses with better value for specific objectives. This is due to a lack of data collection. The courses that we have gathered are not diverse according to the criteria. Both GA and ACO are implemented in the computer with detailed configuration as shown in **Table 1**. We evaluated both of the proposed algorithms on the tested dataset.

TABLE 1. System configuration for experiments.

Item	Info
CPU	Intel(R) Core(TM) i9-9900KF CPU @ 3.60GHz , 12 cores.
RAM	Corsair Vengeance LPX 16GB
Programming Platform	Python 3
Operating System	Window 10

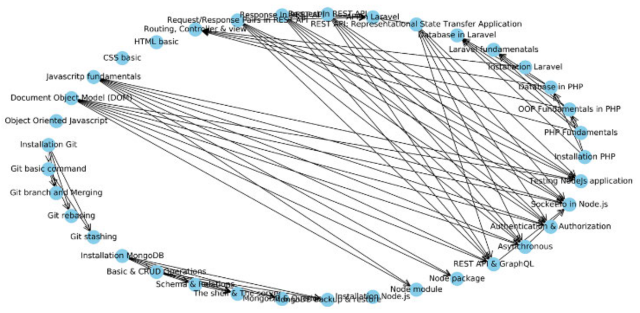

FIGURE 5. Learning outcomes required by the job "Junior Web Dev (Laravel or Nodejs)".

B. RESULTS

A series of parameters govern both GA and ACO. We execute the algorithm several times with the collected dataset to indicate the best set of parameters shown in **Table 2**. Part A of the table displays the parameters to run the GA.

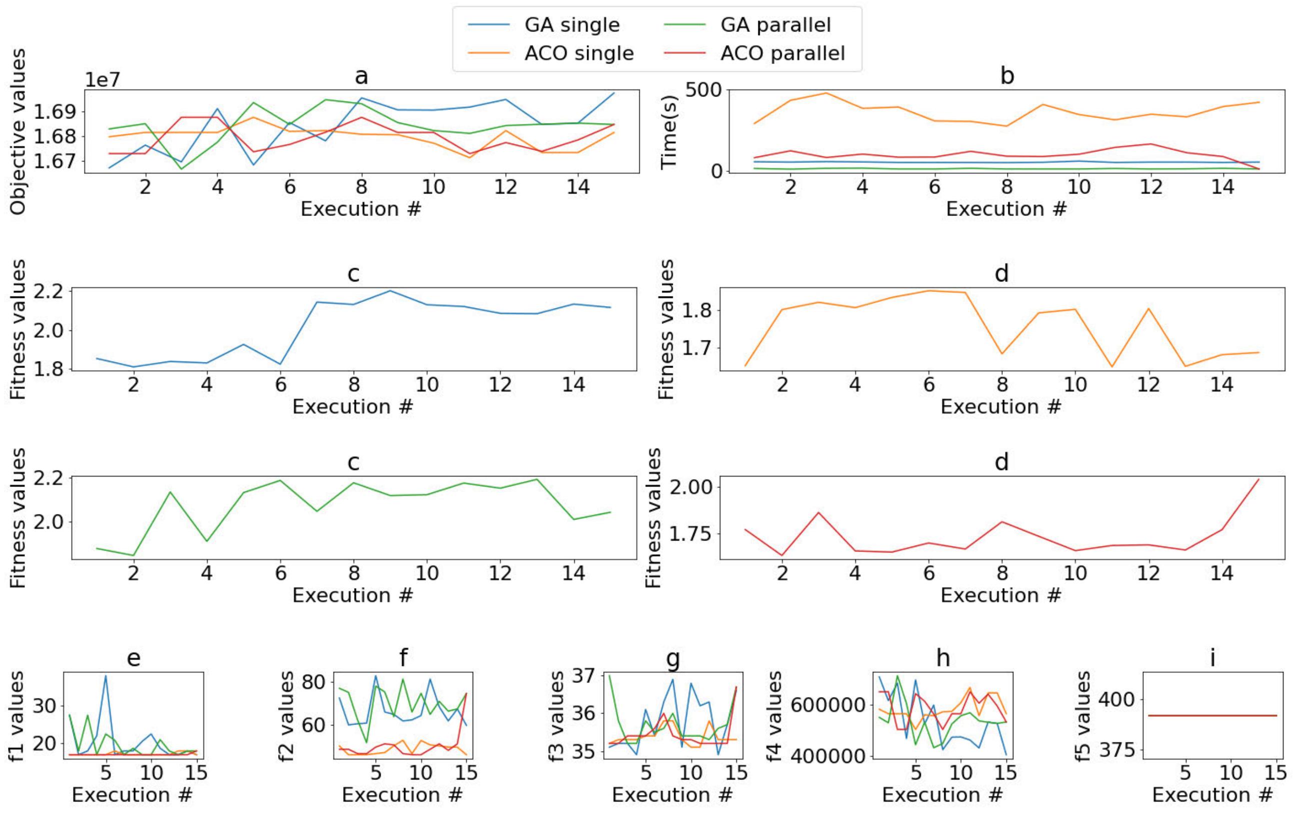

FIGURE 6. Outputs of 15 executions of GA and ACO. a) Objective values generated by different mode of GA and ACO. b) Execution time to complete the execution of different mode GA and ACO. c) Fitness values at the last generation of GA. d) Fitness values at the last loop of ACO. e,f,g,h,i) illustrates the objective values returned by f_1, f_2, f_3, f_4, f_5 to corresponding execution.

TABLE 2. Parameters to execute the algorithms.

A) Obtained Parameters for GA.

Parameter	Value
Number of population $\boldsymbol{\beta}$	0.3 * **C**
Exchange genes rate $\boldsymbol{\eta}$	0.6
Mutation rate δ	0.3
Selection rate $\boldsymbol{\lambda}$	0.1
rate of elites **w**	0.5

B) Obtained Parameters for ACO.

Parameter	Value
Influence of pheromones α	1
Influence of objective β	5
Vaporation rate ρ	0.01
pheromone strength Q	0.0005
number of ants M	200

The remaining part illustrates the parameters of the ACO. Meta-heuristic algorithms do not guarantee to find the optimal solution with different initialization values. Therefore, we run the algorithm 15 times with other initial solutions to find a learning path to archive the job of "Junior Web Dev (Laravel or Nodejs)" for a newbie. **Figure 5** shows the required LO in the knowledge-graph of the tested job. It has been chosen to illustrate algorithms' results because there are a few paths to meet the required learning outcomes.

The outputs of both algorithms over 15 executions for the selected job are shown in **Figure 6**. Both algorithms are designed to prioritize the search for courses that contain more learning outcomes related to queries. It can be observed that the fitness values of both algorithms do not change much over different executions. The objectives values (with $norm(x) = x$) brought between the runs of GA are equivalent to ACO. ACO finds better critical paths (f_1 values) and less learning effort (f_2 values) than GA, while the other indicators are similar. f_5 values are exactly the same because the collected courses are charged 45$ per month. These results show the stability of the algorithms on the selected sets of parameters.

The processing time of ACO (average of 361.8667 seconds) is 6.68 times slower than that of GA (average of 54.86667 seconds). The computation speed is improved significantly when using 6 processing cores for both algorithms. The computation speed increases rapidly while maintaining the solutions' quality. The parallel mode's execution time of the algorithms are improved according to the number of cores used, as shown in **Figure 7**. The graph is not linear because of several steps of 2 algorithms implemented synchronization.

Figure 8 illustrates the change (ability to converge) of Fitness values across the loops to both GA and ACO. We canceled the stop condition of the algorithms so the algorithm can run up to 200 loops. It can be seen that GA archives its best solution after 75 loops. ACO takes 165 loops to gain the

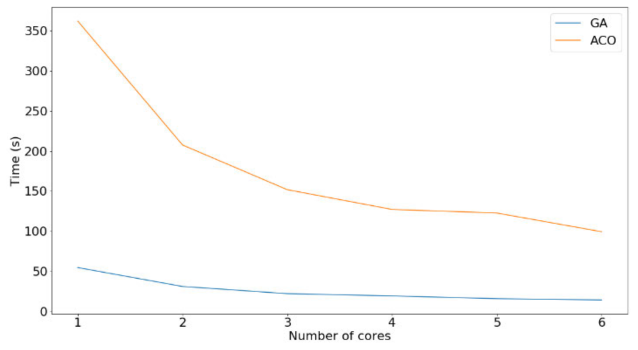

FIGURE 7. The processing speed of the proposed algorithm improves proportionally with the number of processing cores used.

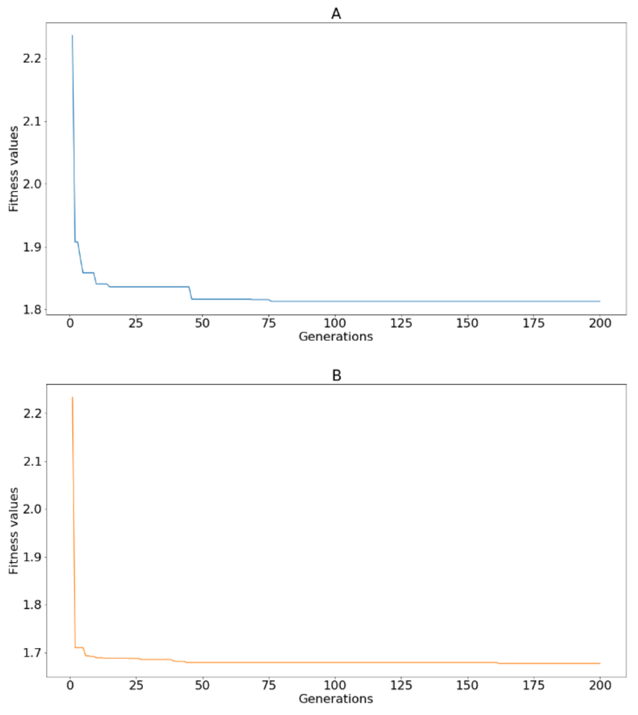

FIGURE 8. Fitness values generated by GA and ACO over generations.

solution. The convergence of both algorithms can be similar (slope of the graph than GA. This leads to its slower execution time. We do not directly compare the values of these two target values. Because even though they use the same norm function mechanism. The ***dump***$_i$ values obtained from the first loop of each algorithm are different. Therefore, the scale of the fitness values is also different. These ranges even vary on different runs.

We illustrate two generated learning paths by GA and ACO graphically in **Figure 9**. The start and end nodes are two artificial courses used to transform the graph into a tree (single rood and single leaf), as mentioned in this paper discussing the critical path computation.

The results of the algorithms vary with each goal to be achieved. We conduct an algorithmic evaluation of 18 jobs as the query. Their results are shown in **Table 3** and **Table 4** (The algorithms are executed in parallel mode). For most of the jobs that contain a single path to be archived, both algorithms provide the same solutions. However, GA was slightly better than ACO when dealing with jobs that exist more than one path. GA has absolute dominance in terms of processing time. The comparison of objective values and execution time between both algorithms in different modes shown in **Figure 10**. It seems that GA's approach of discovery is similar to the tracking possible agent paths in ACO for the tested dataset. When a query contains more than one job, it also means that the search space increases rapidly. We combine jobs to do queries that test both proposed algorithms in different modes. **Tables 5** shows the two algorithms' results when dealing with a query containing 2 to 18 jobs using parallel mode. ACO and GA give a similar quality of the solutions when the query's complexity increases. Although both algorithms having the same mechanism towards courses covering the most LOs tend to be recommended. It leads to GA gives similar objective values to ACO in most cases. The processing time of GA is overwhelming compared to ACO.

This gap is even greater with more complex queries as shown in **Table 5**.

To check if the two algorithms can obtain an optimal solution when the search space is small. We run the test on a smaller dataset. It includes 40 courses adapted for the query of "Junior Web Dev (Laravel or Nodejs)." We compare both GA, ACO with a brute force algorithm installation. **Table 6** illustrates the results of both implementations. The proposed GA and ACO also got an optimal solution. Meantime, the time execution was much better than the

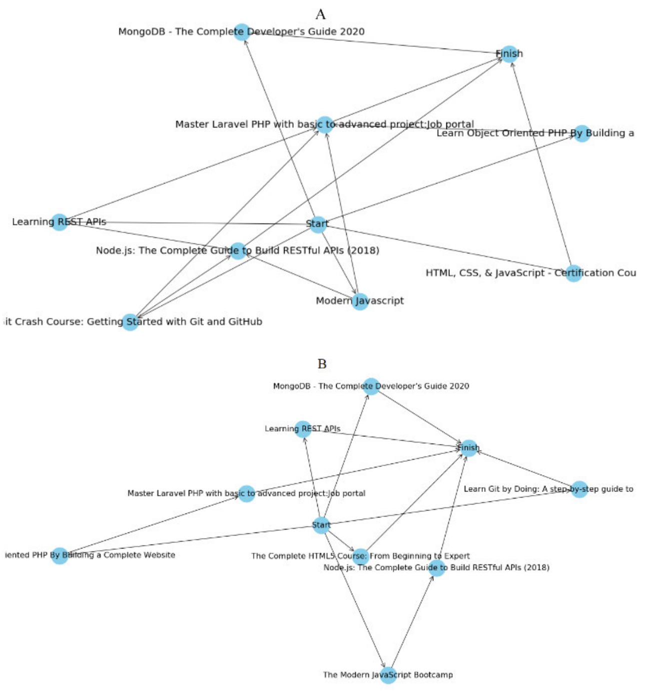
A
MongoDB - The Complete Developer's Guide 2020
Finish
Master Laravel PHP with basic to advanced project:Job portal
Learn Object Oriented PHP By Building a
Learning REST APIs
Start
Node.js: The Complete Guide to Build RESTful APIs (2018)
HTML, CSS, & JavaScript - Certification Cou
Modern Javascript
it Crash Course: Getting Started with Git and GitHub
B
MongoDB - The Complete Developer's Guide 2020
Learning REST APIs
Finish
Master Laravel PHP with basic to advanced project:Job portal
Learn Git by Doing: A step-by-step guide to
Start
iented PHP By Building a Complete Website
The Complete HTML5 Course: From Beginning to Expert
Node.js: The Complete Guide to Build RESTful APIs (2018)
The Modern JavaScript Bootcamp

FIGURE 9. A) learning path generated by GA. B) learning path generated by ACO.

TABLE 3. Outputs of GA for corresponding jobs.

Job	Fitness Value	Objective value	f1	f2	f3	f4	f5	Number of Required LOs	Number of Archived LOs	Time (s)
Python for Data Science	1.7003	16768650	12.76667	21.36667	14.8	611701	121	16	16	3
PHP Developer (Laravel)	2.0209	16959640	12.6	23.9	26.7	420711	294	24	28	15
Fresher Frontend ReactJS Developer	1.9273	17281381	16	19	17	98970	196	12	22	14
ReactJS Developer	1.8338	17253893	10.5	11.5	12.9	126458	147	9	9	13
Junior Web Dev (Laravel or Nodejs)	2.1514	16850780	17	57.4	35.7	529571	392	37	43	15
Backend Developer (Golang/NodeJS)	2.133	16910855	18	77.3	43.8	469496	490	37	39	20
Backend Developer (PHP, Go)	2.1616	16718090	22	106.5666 7	62.3	662261	686	46	57	25
Frontend Dev (ReactJS/JavaScript, HTML5)	1.6381	17030443	14	34.8	26.3	349908	294	19	31	14
Web Developer (Angular, NodeJS, Python)	2.0018	16746465	17	58.5	41.1	633886	441	34	46	16
Jr/Sr Backend NodeJS (JavaScript/MySQL)	2.1201	16662998	21	61.3	44.3	717353	490	40	48	20
Java Core Developer (MySQL,Linux)	2.2361	16515938	66.37	175.12	51.8	864413	588	43	58	23
Senior Java Dev (SQL, Oracle)	1.9951	17303354	66.37	111.74	33.1	76997	392	30	49	22
Java Developer (Java, SQL)	2.0073	16211307	18	84.05	59.9	1169044	686	31	67	27
Java Web Dev (Spring, JavaScript)	1.615	17272697	12.5	31.57	25.4	107654	294	22	31	23
Java dev	2.0691	16220460	24.5	75.32	40.8	1159891	490	46	62	35
Sr Java Dev (3 years Spring)	2.0704	16439912	22.81	70.08	44.4	940439	539	43	72	28
Fullstack Dev (Java, JavaScript)	1.774	16515581	13.15	43.19	31.8	864770	343	21	37	21
Java Developers (JavaScript, Spring)	2.2361	16467632	22.81	71.08	44.8	912719	539	41	70	30

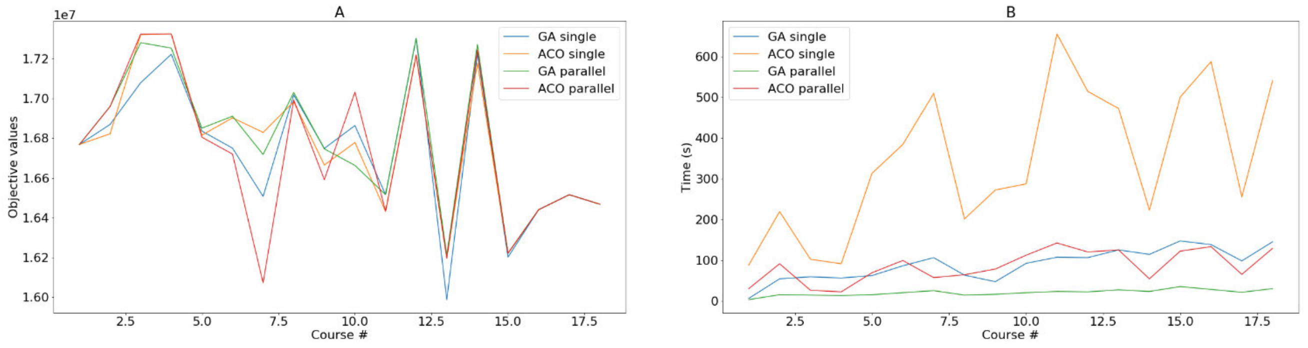

FIGURE 10. Comparison of proposed algorithms outputs on different query. A) Objective value, B) Processing time.

Exhaustive search. All algorithms are executed in parallel mode.

Decision-makers can customize the results according to their preferences by changing the distance function's dimensions' weight parameters. **Table 7** shows the different values of objective functions according to the different sets of parameters. The order of the parameters is listed in the order of objective functions defined in section 3. The order of the parameters is listed in the order of objective functions defined in section 3. The range of values of the values in the 5 dimensions of space for decision making is different. So if using a weighted value that is too small does not make a dif-

TABLE 4. Outputs of ACO for corresponding jobs.

Job	Fitness Value	Objective Value	f1	f2	f3	f4	f5	Number of Required Los	Number of Archived Los	Time (s)
Python for Data Science	1.899	16768650	12.76667	21.36667	14.8	611701	121	16	16	30
PHP Developer (Laravel)	1.7885	16959640	12.6	23.9	26.7	420711	294	24	28	91
Fresher Frontend ReactJS Developer	1.652	17324438	9.5	10.5	17.5	55913	196	12	12	26
ReactJS Developer	1.7677	17325321	7	8	13.5	55030	147	9	9	22
Junior Web Dev (Laravel or Nodejs)	1.6925	16805179	17	46.8	35.3	575172	392	37	37	69
Backend Developer (Golang/NodeJS)	1.6275	16720244	17	71.8	40.6	660107	441	37	38	99
Backend Developer (PHP, Go)	1.7192	16071871	15	88.9	58	1308480	637	46	54	57
Frontend Dev (ReactJS/JavaScript, HTML5)	1.7078	16992984	7.5	16.3	26.3	387367	294	19	20	64
Web Developer (Angular, NodeJS, Python)	1.6838	16591531	31	66.3	35.9	788820	392	34	37	78
Jr/Sr Backend NodeJS (JavaScript/MySQL)	1.6339	17031970	17	46.95	40.4	348381	441	40	44	112
Java Core Developer (MySQL,Linux)	2.1733	16431509	32.29	131.09	56.4	948842	637	43	68	142
Senior Java Dev (SQL, Oracle)	2.0788	17218925	29.57	67.71	37.7	161426	441	30	59	120
Java Developer (Java, SQL)	1.853	16195675	17.63	64.07	47.6	1184676	588	31	53	125
Java Web Dev (Spring, JavaScript)	1.6785	17244099	7.5	15.07	25.4	136252	294	22	26	54
Java dev	1.8122	16220460	24.5	75.32	40.8	1159891	490	46	62	122
Sr Java Dev (3 years Spring)	1.7715	16439912	22.81	70.08	44.4	940439	539	43	72	133
Fullstack Dev (Java, JavaScript)	1.8871	16515581	13.15	43.19	31.8	864770	343	21	37	65
Java Developers (JavaScript, Spring)	1.7671	16467632	22.81	71.08	44.8	912719	539	41	70	129

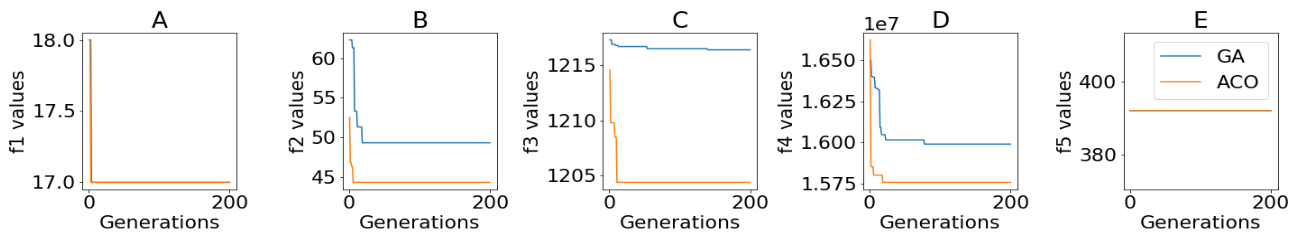

FIGURE 11. Single-objective values changed over loops of the GA and ACO.

ference. The available learning paths are suitable for testing the "Python for Data Science" job. We added some similar data to the original data. The additional data have similar LOs to some available courses in the recommended learning paths. Still, different values for each property to illustrate how changing the weight affect the algorithm's results.

We change the proposed objective function by these original functions. We can identify the Pareto points to compare with archiving solutions from the proposed approach. **Figure 11** illustrates the solutions archived from 2 algorithms through each of the original target functions. Our implemented GA showed superiority when the last generation's results were equal to or better than ACO implementation.

Compared with previous studies, our model is complete and more applicable for MOOC [14], [15]. Out problem requires the mapping from learning outcome to courses increases the problem's difficulty by one level compared to the previously proposed models. We consider more user's demands. Our mathematical model is a more detailed guideline to set up in different optimization solvers.

TABLE 5. Outputs of algorithms for complex queries contains multiple jobs.

A) OUTPUTS OF GA

Number of jobs in Query	Fitness Value	obj	f1	f2	f3	f4	f5	Number of Required LOs	Number of Archived LOs	Time (s)
2	1.7131	16194239	20.5	79.43334	42.2	1186112	441	39	46	14
3	1.8685	16433099	30	117.83334	49	947252	539	46	52	17
4	2.1441	16585418	28	113.66667	50.8	794933	476	46	59	17
5	2.1633	16153302	36.5	135.46667	63.7	1227049	686	65	72	28
6	2.207	15795550	38.6	187.26667	82.5	1584801	882	77	86	27
7	2.0591	15929328	36.5	181.36667	91.6	1451023	980	85	99	27
8	2.0268	16276260	30	196.66667	86.2	1104091	980	85	95	24
9	2.0384	16028844	23.1	187.86667	94.9	1351507	1003	86	96	43
10	2.1651	15983526	38	178.36667	94.2	1396825	1029	86	98	29
11	2.1564	15336514	81.37	380.97334	135	2043837	1444	129	137	55
12	2.2116	15151900	82.37	414.98334	137.6	2228451	1519	129	137	37
13	2.2159	15508918	82.37	422.08667	139.3	1871433	1453	130	138	42
14	2.2336	15482969	88.53	380.75	135.7	1897382	1519	130	142	37
15	2.1021	14905999	47.95	392.16667	159	2474352	1715	150	157	70
16	2.0951	15007044	46.5	405.17334	153.3	2373307	1666	152	165	47
17	2.0199	14944731	39	389.40667	153.4	2435620	1600	153	166	64
18	2.1171	15272292	42.5	441.67334	160.8	2108059	1764	153	165	54

B) OUTPUTS OF ACO

Number of jobs in Query	Fitness Value	*obj*	f1	f2	f3	f4	f5	Number of Required LOs	Number of Archived LOs	Time (s)
2	1.6638	16198083	12.76667	51.26667	40.9	1182268	415	39	42	143
3	1.6908	16206523	16.5	69.76667	50.2	1173828	513	46	57	141
4	1.7416	16243752	15	65.26667	49.8	1136599	513	46	52	140
5	1.8132	16362391	18	104.16667	63.6	1017960	660	65	71	111
6	1.7849	16133212	17	129.76667	82	1247139	856	77	83	175
7	1.7617	16233987	18	152.76667	90.9	1146364	954	85	92	194
8	1.7085	16275662	17	144.76667	90.4	1104689	980	85	101	172
9	1.7486	15893252	18	149.36667	91	1487099	954	86	101	214
10	1.8288	16023704	17.1	151.36667	91.4	1356647	954	86	101	184
11	1.8827	15072862	32.29	318.13334	143	2307489	1542	129	155	202
12	1.8923	14930629	32.29	315.85667	147	2449722	1591	129	149	222
13	1.8481	15425400	32.29	287.18667	146.9	1954951	1591	130	150	250
14	1.8596	15073632	32.29	309.47667	140.8	2306719	1542	130	148	332
15	1.8225	15095160	32.29	349.86667	151.9	2285191	1629	150	172	430
16	1.9134	15037869	32.29	330.19667	151.6	2342482	1640	152	171	731
17	1.8632	15070955	32.29	315.77667	150.7	2309396	1666	153	165	415
18	1.9158	15054673	32.29	306.66667	155.1	2325678	1689	153	165	374

V. CONCLUSION

This study developed a multi-objective optimization model as a knowledge-based recommender for MOOC's learning path. The model is generic for learning paths and can be applied in any online learning framework compared to other proposed methods. It allows decision-maker to assign their

TABLE 6. Obtained results after executing proposed algorithms and exhaustive search algorithm.

Algorithm	Brute Force	GA	ACO
Objective Value	892315.11	892315.11	892315.11
Execution Time(s)	1688	34	162

TABLE 7. Different set of parameters and corresponding objective values.

A) OBJECTIVE VALUES RETURNED BY GA

Set of Parameters	f_1	f_2	f_3	f_4	f_5
w_1 =1000; w_2 =1; w_3 =1; w_4 =1; w_5 =1	**17**	60.3	36.3	646447	392
w_1 =1; w_2 =1000; w_3 =1; w_4 =1; w_5 =1	38	**54.3**	35.9	497911	392
w_1 =1; w_2 =1; w_3 =1000; w_4 =1; w_5 =1	54.5	124.3	**41.8**	507005	441
w_1 =1; w_2 =1; w_3 =1; w_4 =1000; w_5 =1	63	117	35.2	**1366909**	392
w_1 =1; w_2 =1; w_3 =1; w_4 =1; w_5 =1000	80.5	168.3	36	1131613	**392**

B) OBJECTIVE VALUES RETURNED BY ACO

Set of Parameters	f_1	f_2	f_3	f_4	f_5
w_1 =1000; w_2 =1; w_3 =1; w_4 =1; w_5 =1	**17**	46.3	35.3	565951	392
w_1 =1; w_2 =1000; w_3 =1; w_4 =1; w_5 =1	17	**43.3**	39.2	543063	441
w_1 =1; w_2 =1; w_3 =1000; w_4 =1; w_5 =1	95.03	177.98	**54.1**	1206616	588
w_1 =1; w_2 =1; w_3 =1; w_4 =1000; w_5 =1	107.53	225.48	49.7	**1919559**	539
w_1 =1; w_2 =1; w_3 =1; w_4 =1; w_5 =1000	17	58.3	36.6	504258	**392**

preferences to achieve different goals of the recommended path. The model operates based on the knowledge-graph and offers recommendations close to the user's goal, but the knowledge- graph has not been easily constructed. The indicating of learning outcome inter-dependencies is only favorable in the course of course development. In the context of so many courses developed, there is a need to effectively explore the learning outcomes being stored in complex structures. This can be done manually, but it certainly costs a lot. Several researchers are looking to explore the dependency of learning content in e-learning systems. Pan *et al.* introduced the graph-based propagation algorithm to retrieve the prerequisite relation between knowledge concepts [28]. Liu *et al.* propose an approach to concept graph learning from different providers [29]. Cheng *et al.* developed a system to construct the graph of knowledge based on deep learning techniques [30].

We also develop versions of the GA and ACO. They allow us to effectively solve the problem to provide the runtime response for a single request. The experiments' results show the propose d algorithm's efficiency with test-data, where GA gained a slightly better result than the ACO. However, we need to test other research directions to improve the quality of the proposed algorithms. Some potential research directions include but are not limited to Chaotic Random Spare ACO [31], another EA approach like Evolutionary biogeography-based [32], hybrid method for EA [33]. Our future work is also to build a model that

TABLE 8. Parallel scheme of GA.

```
1:  Do in sequence
2:    Step 1: Initialize the first population $g = 1$
3:      While $i < £$
4:        Add the task of individual $\theta_i^{(g)}$ to queue $Q_{init}$.
5:        $i = i + 1$.
6:      Do in parallel (for each core)
7:        If $Q_{init}$ is not empty.
8:          Initialize the individual $\theta_t^{(g)}$ where:
            $t = dequeue(Q_{init})$
9:    Step 2: Selection
10:     Select $£ * \gamma$ best individuals to the new generation.
11:   Step 3: Generate the new generation.
11:     Select the parents.
12:     Add the task of create individuals to queue $Q_{cross}$.
13:     Do in parallel (for each core)
14:       Do in sequence
15:         If $Q_{cross}$ is not empty
16:           Crossover the parents to create
              individual $\theta_t^{(g)}$ corresponding to
              $t = dequeue(Q_{init})$.
17:           Mutation individual $\theta_t^{(g+1)}$.
18:           Repair individual $\theta_t^{(g+1)}$.
19:           Compute fitness values of $\theta_t^{(g+1)}$
20:   Step 4: Check the algorithm convergence.
21:     If convergence condition satisfied.
22:       Stop algorithm.
23:     Otherwise, $g = g + 1$ comeback step 2.
```

TABLE 9. Parallel scheme of ACO.

```
1:  Do in sequence
2:    Step 1: Initialize the first population $g = 1$
3:      While $i < £$
4:        Add the task of ant $i^{th}$ to queue $Q_{init}$.
5:        $i = i + 1$.
6:      Do in parallel (for each core)
7:        Do in sequence
8:          If $Q_{init}$ is not empty
9:            $t = dequeue(Q_{init})$
10:           Update the trail for ant $t^{th}$.
11:           Compute the fitness value for ant $t^{th}$.
12:   Step 2: Update the pheromones.
13:   Step 3: Check the algorithm convergence.
14:     If convergence condition satisfied.
15:       Stop algorithm
16:     Otherwise, $g = g + 1$ comeback step 2.
```

allows the personalization of learners' learning paths based on demand and similarity among learners, user feedback, etc. Although the main objectives were not attached to the career of the learning path recommendation at MOOC problem, Shi *et al.* used more demographic info to support the quality of recommendation, which is also an exciting research direction [15]. Today, many parallel computing applications use the GPU for computation instead of CPU, like what they are doing. We plan to develop the algorithm to run on GPU to archive the maximum speed improvement [34].

APPENDIX

See Tables 8 and 9.

REFERENCES

[1] I. Aguaded, "The MOOC revolution: A new form of education from the technological paradigm?" *Comunicar*, vol. 21, no. 41, pp. 7–08, Jun. 2013, doi: 10.3916/c41-2013-a1.

[2] R. Burke, "Knowledge-based recommender systems," in *Encyclopedia of Library and Information Sciences*, vol. 69, no. 32. Berlin, Germany: Springer, 2000, pp. 180–200.

[3] N. T. Son, D. H. Dat, N. Q. Trung, and B. N. Anh, "Combination of dimensionality reduction and user clustering for collaborative-filtering," in *Proc. Int. Conf. Comput. Sci. Artif. Intell. (CSAI)*, 2017, pp. 125–130, doi: 10.1145/3168390.3168405.

[4] L.-H. Wong and C.-K. Looi, "Adaptable learning pathway generation with ant colony optimization," *Educ. Technol. Soc.*, vol. 12, no. 3, pp. 309–326, 2009.

[5] T.-C. Hsieh and T.-I. Wang, "A mining-based approach on discovering courses pattern for constructing suitable learning path," *Expert Syst. Appl.*, vol. 37, no. 6, pp. 4156–4167, Jun. 2010.

[6] I. Jugo, B. Kovacic, and V. Slavuj, "Using data mining for learning path recommendation and visualization in an intelligent tutoring system," in *Proc. 37th Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, May 2014, pp. 924–928, doi: 10.1109/mipro.2014.6859700.

[7] V. Tam, E. Y. Lam, and S. T. Fung, "A new framework of concept clustering and learning path optimization to develop the next-generation e-learning systems," *J. Comput. Educ.*, vol. 1, no. 4, pp. 335–352, Dec. 2014, doi: 10.1007/s40692-014-0016-8.

[8] C.-M. Chen, "Intelligent Web-based learning system with personalized learning path guidance," *Comput. Educ.*, vol. 51, no. 2, pp. 787–814, Sep. 2008.

[9] T. Tang and G. McCalla, "Data mining for contextual educational recommendation and evaluation strategies," in *Chapman & Hall/CRC Data Mining and Knowledge Discovery Series*. Boca Raton, FL, USA: CRC Press, 2010, pp. 257–272, doi: 10.1201/b10274-21.

[10] M.-H. Hsu, "A personalized english learning recommender system for ESL students," *Expert Syst. Appl.*, vol. 34, no. 1, pp. 683–688, Jan. 2008, doi: 10.1016/j.eswa.2006.10.004.

[11] E. Kurilovas, I. Zilinskiene, and V. Dagiene, "Recommending suitable learning scenarios according to learners' preferences: An improved swarm based approach," *Comput. Hum. Behav.*, vol. 30, pp. 550–557, Jan. 2014, doi: 10.1016/j.chb.2013.06.036.

[12] S. Wan and Z. Niu, "A learner oriented learning recommendation approach based on mixed concept mapping and immune algorithm," *Knowl.-Based Syst.*, vol. 103, pp. 28–40, Jul. 2016.

[13] V. Shmelev, M. Karpova, and A. Dukhanov, "An approach of learning path sequencing based on revised Bloom's taxonomy and domain ontologies with the use of genetic algorithms," *Procedia Comput. Sci.*, vol. 66, pp. 711–719, Jan. 2015, doi: 10.1016/j.procs.2015.11.081.

[14] H. Zhu, F. Tian, K. Wu, N. Shah, Y. Chen, Y. Ni, X. Zhang, K.-M. Chao, and Q. Zheng, "A multi-constraint learning path recommendation algorithm based on knowledge map," *Knowl.-Based Syst.*, vol. 143, pp. 102–114, Mar. 2018, doi: 10.1016/j.knosys.2017.12.011.

[15] D. Shi, T. Wang, H. Xing, and H. Xu, "A learning path recommendation model based on a multidimensional knowledge graph framework for e-learning," *Knowl.-Based Syst.*, vol. 195, May 2020, Art. no. 105618, doi: 10.1016/j.knosys.2020.105618.

[16] S. Al-Muhaideb and M. E. B. Menai, "Evolutionary computation approaches to the curriculum sequencing problem," *Natural Comput.*, vol. 10, no. 2, pp. 891–920, Jun. 2011.

[17] K. Sorensen, M. Sevaux, and F. Glover, "A history of metaheuristics," 2017, *arXiv:1704.00853*. [Online]. Available: http://arxiv.org/abs/1704.00853

[18] Q. Zhao, Y. Zhang, and J. Chen, "An improved ant colony optimization algorithm for recommendation of micro-learning path," in *Proc. IEEE Int. Conf. Comput. Inf. Technol. (CIT)*, Dec. 2016, pp. 190–196, doi: 10.1109/CIT.2016.47.

[19] C.-L. Hwang and A. S. Md Masud, *Multiple Objective Decision Making, Methods and Applications: A State-of-the-Art Survey*. Berlin, Germany: Springer-Verlag, 1979.

[20] C. Romero and T. Rehman, Eds., "Chapter five compromise programming," in *Developments in Agricultural Economics*, vol. 11. Amsterdam, The Netherlands: Elsevier, 2003, pp. 63–78. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0926558903800079, doi: 10.1016/S0926-5589(03)80007-9.

[21] T. S. Ngo, N. A. Bui, T. T. Tran, P. C. Le, D. C. Bui, T. D. Nguyen, L. D. Phan, Q. T. Kieu, B. S. Nguyen, and S. N. Tran, "Some algorithms to solve a bi-objectives problem for team selection," *Appl. Sci.*, vol. 10, no. 8, p. 2700, Apr. 2020.

[22] S. T. Ngo, J. Jaafar, I. A. Aziz, and B. N. Anh, "A compromise programming for multi-objective task assignment problem," *Computers*, vol. 10, no. 2, p. 15, Jan. 2021, doi: 10.3390/computers10020015.

[23] R. Li, Y. Leung, H. Lin, and B. Huang, "An adaptive compromise programming method for multi-objective path optimization," *J. Geographical Syst.*, vol. 15, no. 2, pp. 211–228, Apr. 2013, doi: 10.1007/s10109-012-0172-1.

[24] C. A. C. Coello, G. B. Lamont, and D. A. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York, NY, USA: Springer, 2007.

[25] V. V. Vazirani, "Minimum makespan scheduling," in *Approximation Algorithms*. Berlin, Germany: Springer, 2003, doi: 10.1007/978-3-662-04565-7_10.

[26] M. Nicolas, G. Frédéric, and S. Patrick, *Artificial Ants*. Hoboken, NJ, USA: Wiley, 2010.

[27] H. Rocha, I. S. Peretta, G. F. M. Lima, L. G. Marques, and K. Yamanaka, "Exterior lighting computer-automated design based on multi-criteria parallel evolutionary algorithm: Optimized designs for illumination quality and energy efficiency," *Expert Syst. Appl.*, vol. 45, pp. 208–222, Mar. 2016.

[28] L. Pan, X. Wang, C. Li, J. Li, and J. Tang, "Course concept extraction in moocs via embedding-based graph propagation," in *Proc. 8th Int. Joint Conf. Natural Lang. Process.*, 2017, pp. 875–884.

[29] H. Liu, W. Ma, Y. Yang, and J. Carbonell, "Learning concept graphs from online educational data," *J. Artif. Intell. Res.*, vol. 55, pp. 1059–1090, Apr. 2016.

[30] P. Chen, Y. Lu, V. W. Zheng, X. Chen, and B. Yang, "KnowEdu: A system to construct knowledge graph for education," *IEEE Access*, vol. 6, pp. 31553–31563, 2018, doi: 10.1109/access.2018.2839607.

[31] D. Zhao, L. Liu, F. Yu, A. A. Heidari, M. Wang, G. Liang, K. Muhammad, and H. Chen, "Chaotic random spare ant colony optimization for multi-threshold image segmentation of 2D kapur entropy," *Knowl.-Based Syst.*, vol. 216, Mar. 2021, Art. no. 106510, doi: 10.1016/j.knosys.2020.106510.

[32] J. Tu, H. Chen, J. Liu, A. A. Heidari, X. Zhang, M. Wang, R. Ruby, and Q.-V. Pham, "Evolutionary biogeography-based whale optimization methods with communication structure: Towards measuring the balance," *Knowl.-Based Syst.*, vol. 212, Jan. 2021, Art. no. 106642, doi: 10.1016/j.knosys.2020.106642.

[33] M. A. Al-Furhud and Z. H. Ahmed, "Experimental study of a hybrid genetic algorithm for the multiple travelling salesman problem," *Math. Problems Eng.*, vol. 2020, pp. 1–13, Oct. 2020, doi: 10.1155/2020/3431420.

[34] M. L. Wong and G. Cui, "Data mining using parallel multiobjective evolutionary algorithms on graphics processing units," in *Massively Parallel Evolutionary Computation on GPGPUs*. Berlin, Germany: Springer, 2013, pp. 287–307.

NGO TUNG SON received the bachelor's degree in computing, in 2011, where he is currently pursuing the master's degree in computer science from Vietnam-France University, Vietnam. He is also pursuing the Ph.D. degree in information technology with Universiti Teknologi PETRONAS (UTP), Malaysia. From 2014 to 2016, he worked with the Panasonic Research and Development Center, Hanoi, Vietnam, as a Software Engineer. Since 2016, he worked with FPT University, as a Lecturer in information technology. His research interests include artificial intelligence, adaptive and intelligent systems, decision support systems, and data mining.

JAFREEZAL JAAFAR (Senior Member, IEEE) received the B.Sc. degree in computer science from Universiti Teknologi Malaysia, in 1998, the M.App.Sc. degree in IT from RMIT University, Australia, in 2002, and the Ph.D. degree from the University of Edinburgh, U.K., in 2009. He joined the Department of Computer and Information Sciences, Universiti Teknologi PETRONAS (UTP), Malaysia, in 1999, where he was appointed as the Head, from 2012 to 2016. He was the former Head of the Center for Research in Data Science, from 2017 to 2020, which focuses on implementation of machine learning and predictive analytic in oil and gas (O&G) industry. He is currently an Associate Professor and the Dean of the Faculty of Science and Information Technology, UTP. He is also active in conducting professional training in AI and big data analytic for the public and industry. His main research interests include AI, machine learning, and data analytics, with over 100 technical publications. Based on his experience and expertise, he been awarded as the Professional Technologies (P.Tech.) by Malaysia Board of Technology, a member of the Academy Science Malaysia SIG in Machine Learning, a Senior Member and the Executive Committee for IEEE CS Malaysia Chapter, from 2016 to 2018, and the Executive Committee for MyAIS, from 2017 to 2019.

IZZATDIN ABDUL AZIZ (Member, IEEE) received the master's degree in information technology specializing in computer networks from the University of Sydney, Australia, and the Ph.D. degree in information technology from Deakin University, Australia, working in the domain of data-driven hydrocarbon exploration and cloud computing. He is currently an Associate Professor (AP) and a Principal Researcher with the Center for Research in Data Science (CeRDaS), Universiti Teknologi PETRONAS (UTP) focuses on solving complex upstream oil and gas (O and G) industry problems from the viewpoint of computer sciences and data analytics. He has secured and delivered numerous computing-related O and G projects mainly relating to seismic data processing and workflows for processing hydrocarbon exploration datasets on cloud-based systems. He is also actively working on real industrial projects to predict machine failure at oil and gas platforms and refineries through machine learning approaches. Most of his research is mainly funded by the Petroleum Research Fund, PETRONAS Yayasan UTP fund, and The Malaysian Ministry of Higher Education.

BUI NGOC ANH received the bachelor's and master's degrees in computer science from the Hanoi University of Technology, in 2002 and 2006, respectively. Since 2003, he has been worked on various projects in the software industry. With nearly 20 years of experience in the software industry, he is one of the experts with FPT Technology Corporation, Vietnam. In 2011, he started his work as an IT Lecturer with FPT University, Vietnam. Since 2019, he is the Head of Computing Fundamentals Department, FPT University. He is also the Leader of the SAP-LAP FPT Laboratory, FPT University. His research interests include big data, data mining, computer vision, and software engineering.