Genetic Algorithm for Solving Multi-Objective Optimization in Examination Timetabling Problem

https://doi.org/10.3991/ijet.v16i11.21017

Son Ngo Tung (✉),
Universiti Teknologi Petronas, Seri Iskandar, Malaysia
FPT University, Hanoi, Vietnam
sonnt69@fe.edu.vn

Jafreezal B Jaafar, Izzatdin Abdul Aziz
Universiti Teknologi Petronas, Seri Iskandar, Malaysia

Hoang Giang Nguyen, Anh Ngoc Bui
FPT University, Hanoi, Vietnam

Abstract—Examination timetabling is one of 3 critical timetabling jobs besides enrollment timetabling and teaching assignment. After a semester, scheduling examinations is not always an easy job in education management, especially for many data. The timetabling problem is an optimization and Np-hard problem. In this study, we build a multi-objective optimizer to create exam schedules for more than 2500 students. Our model aims to optimize the material costs while ensuring the dignity of the exam and students' convenience while considering the design of the rooms, the time requirement of each exam, which involves rules and policy constraints. We propose a programmatic compromise to approach the maximum target optimization model and solve it using the Genetic Algorithm. The results show the effective of the introduced algorithm.

Keywords—Examination timetabling, multi-objective optimization, combinatory optimization, genetic algorithm

1 Introduction

Timetabling problems arise in various forms including educational timetabling, sport timetabling, transportation timetabling. In the training institutions, Timetabling is a difficult process faced every semester. It basically is arranging timeslot for a resource such as students, classes, lectures. The timetabling problems classified into groups:

- University Course Timetabling: Schedule courses into timeslots and assign students, time, rooms to each course. Babaei et al conducted a survey on the problem in 2015 [1]. Ngo et al proposed a multi-objective optimization for solving the teaching task assignment [2]. Ibrahim et al. introduced an approach for the course timetabling problem based on instructors' preferences [3].
- Examination Timetabling: Examination timetabling is one of the most important administrative activities that take place in all academic institutions. It aims to allocate students to exams with limited resource. Qu et al conducted a survey trend and development of techniques of the problem [4].

This research focuses on examination timetabling. A non-trivial timetabling problems are normally NP-Hard problems [5]. This kind of problem is not always possible to reach one global optimal solution. With the increase in the size of resources, the complexity of the problem also increasing, which makes it unfavorable for a systematic approach. Through the literature, there some common constrains that remain after different research. The most common constrains for university exam schedule are:

- Hard constrains: Examination timetabling problem is assigning exams to examination periods and rooms so that the following constraints are respected.

1. Only one exam can be placed in a room at any period.
2. A room cannot be used during periods during which it is not available.
3. An exam must be placed in a room (or a set of rooms) so that the overall seating capacity of the rooms equals or is greater than the number of students attending the exam, concerning the requested seating type (e.g., each room has a normal seating and examination seating capacities defined, an exam can request either normal or examination seating). A maximal number of rooms into which an exam can be split cannot be exceeded as well.
4. An exam cannot be placed in a marked period as prohibited for the exam or required if there is some other period required by the exam. Similarly, an exam cannot be placed in a prohibited room for the exam (by the room requirements set on the exam).
5. Required distribution constraints must be satisfied.

- Soft constrains / Objectives: Besides searching for an optimal solution that satisfies all hard constraints mentioned above, the following criteria are optimized.

1. Student preferences: For example: Exam time is continuous or too far away may affect students.
2. Resources penalty: Uneven exploitation of resources leads to resources being overloaded, while others are in a wasted state.

The above constraints are not necessarily present in all situations. Different problem variations may require different settings. In this study, we built an optimizer capable of optimizing three goals: minimizing the difference in the number of students when using the exam room, minimization of the number of exam rooms open, and maximize the student waiting time. While maintaining the hard constraints.

1.1 Related researches

In the past, many researchers study examination timetabling. Our literature focuses on three common aspects: the form of the optimal model for examination timetabling, the constraints as well as the target function used, the optimal problem-solving method. Researchers commonly use the Integer Programming model to descript the timetabling problem. For instance, Liyana and Aizam designed an integer linear Programming base on the preferences and demands obtained through three universities' survey in Malaysia's cost: UMT, UMP, and UMK [6]. Acostamado et al. proposed a mathematical model for the academic timetabling problem and the influence of the system's parameters on the model's size [7]. A non-linear Integer Programming has also been found in [8].

With different test organizations, there will be different requirements and constraints, which will influence the model of the problem that researchers have to take a concern. Dener defined some specific hard constraints relate to the maximum number of exams a student can enter per day, the time between each exam, and some soft constrain relate to the number of students in the same session, and the cost of the resource [9]. Özcan and Ersoy summarized several common constrain to the exam timetabling problem, including exclusions, presets, edge constraints, ordering constraints, event-spread constraints, and attribute constraints [10]. McCollum et al. applied the penalty method, where each violated the soft constraints the receive a reduction in the fitness [8]. There were some other exam schedules, but most of their constraints are common to each other [11],[12]. There some other factors that can be influent in the problem's constraints such as the maximum exams a supervisor can take, number part of the day a student go to exam [13].

Because of the disparity in constraints, there is no inclusive solution and formulation for the exam schedule. This diversity creates different approaches to solve the problem. For instance, Yan and Yu [14] used the hybridization method of Simulated Annealing with multi-neighborhood to improve the Simulated Annealing algorithm's speed. Ender and Ersoy [10] used the Memetic Algorithm to solve the examination timetable problem. The Memetic algorithm is a combination of Genetic Algorithms and hill-climbing. The algorithm uses Genetic Algorithms to create the individual, then calculate the individual's fitness using hill climbing. While using hill-climbing may produce more hard constraints violations case, in return, an individual's overall fitness will be improved. Ayob and Jaradat [15] used Ant Colony Optimization combine with Simulated Annealing and Tabu Search as a local routines searcher. Mandal and Kahar [16] proposed a model using a great deluge algorithm. Murat Dener solved exam scheduling problems in central exams using 2 phases of Genetic Algorithms [9]. One for the assigned courses into sessions, one for assigned students into courses. Shatnawi et al. designed the hybrid of greedy algorithms and genetic algorithms [11]. The greed was used to generate the set of the initial population for the genetic algorithms. Yang et al also introduced an improved Genetic Algorithm for designing their timetable [17]. Bashar et al. proposed a metaheuristic population-based algorithm Intelligent Water Drops Algorithm [18]. However, the solution to this algorithm was not on par with other algorithms. Aside from Evolution algorithms and Swamp algorithms, a graph edge coloring algorithm to search for the timetable had been implemented by Rakesh [19]. The algorithm is split into 2 parts, the first use a bipartite to get a daily schedule in the matrix, then the second phrase assign lectures into the timeslot. Wang designed a genetic algorithm to solve the enrollment timetabling problem [20]. The author used the combination of time, teacher, and course number as the gene coding, the weekly course schedule of each class was a chromosome, the course schedule of the entire school was the initial population. Finally, the fitness was designed according to each class's priority, curriculum dispersion, and teacher satisfaction.

Researchers favor metaheuristic algorithms because of the problem's complexity. In this study, we chose to use Genetic algorithms, a metaheuristic algorithm. Since we want to focus on optimize the soft constrain of the problem, we combine the algorithm with the greed algorithm to create the feasible first population. As for our study's objective, because we found that there are common objectives related to idle-time between exams in other research. However, with different university, their requirement is different from each other. Therefore, we chose to use implement the idle-time relate objective as a flexible function to easily change the content of this objective to satisfy different requirements. Because our study wants to consider the exams' fairness, so we decide to minimize the amount difference of students between exams of the same subject. This objective will balance the burden of each supervisor.

1.2 Contributions of this research

In this study, we present an approach to construct a timetable for the multi-objective schedule problem. Our model aims to maximize the benefit of the organization and the dignity of the exam. We use a combination of linear scalarizing and compromise programming for the proposed multi-objective problem. The model is described in the second section, together with the approach to the multi-objective optimization problem. Section 3 of the paper describes a scheme of the Genetic algorithm for solving the proposed model. We test the proposed model and algorithm by scheduling FPT University students in the spring semester of 2020. The test results are shown in section 4 of the paper. The rests are discussions and conclusions.

2 Problem Formulation

2.1 MOP for examination timetabling

There are some denotations:

- S is the number of subjects.
- M is the number of students.
- R is the number of class rooms.
- ∂ denote the minimum number of students per class
- T denotes the number of time-slots.

- F is the number of conditions of the subjects.
- $I = \{i_r | i_r > 0, i_r \in N, r = 1 \dots R\}$. Where I_r denotes number of students that room r^{th} can contain.
- $J = \{j_s | j_s > 0, j_s \in N, s = 1 \dots S\}$. Where J_s denotes the maximum number of students that an exam subject s^{th} can have.
- $L = \{l_{s,f} | l_{s,f} \in \{0,1\}, s = 1 \dots S, f = 1 \dots F\}$ denotes the required conditions for subjects. Where: $l_{s,f} = 1$ means the subject s^{th} requires condition f^{th}. 0 otherwise.
- $U = \{u_{r,f} | u_{s,f} \in \{0,1\}, r = 1 \dots R, f = 1 \dots F\}$ denotes the conditions for corresponding class-rooms. Where: $u_{s,f} = 1$ means the room r^{th} satisfy the the condition f^{th}. 0 otherwise.
- $A = \{a_{m,s} | \, a_{m,s} = 0 \dots 1, m = 1 \dots M, s = 1 \dots S\}$ denote the enrollment of student. Where: $a_{m,s} = 0$ if the student m^{th} is not going to exam subject s^{th}. 1 otherwise.
- $P = \{p_s | p_s > 0, \; p_s \in N, s = 1 \dots S\}$. Denote the number of slots that subject s^{th} take.
- Decision variables $C = \{C_s | C_s \geq 0, C_s \in \mathbb{N}, s = 1 \dots S\}$ denotes the number of subject s^{th}'s exams.
- Decision Variables $X = \{x_{m,s,c} | x_{m,s,c} \in \{0,1\}, m = 1 \dots M, s = 1 \dots S, c = 1 \dots C_s\}$ Where $x_{m,s,c} = 1$ if the student m^{th} is assigned to the examination zoom c^{th} for the subject s^{th}. $x_{m,s,c} = 0$ otherwise.
- Decision Variables $Y = \{y_{c,s,t,r} | y_{c,s,t,r} \in \{0,1\}, t = 1 \dots T, r = 1 \dots R, s = 1 \dots S, c = 1 \dots C_s\}$ Where $y_{c,s,t,r} = 1$ if the examination c^{th} of the subject s^{th} occupy room r^{th} at the time-slot t^{th}. $y_{c,s,t,r} = 0$ otherwise.
- Decision Variables $Z = \{z_s | z_s > 0, z_s \in N, s = 1 \dots S\}$. Where z_s denotes the slot that examination of subject s^{th} begin.

There are 3 objectives:

- The number of students in the same examination zoom is not too different:

$$min\left(\left|\sum_{m=1}^{M} x_{m,s,c} - \frac{\sum_{m=1}^{M}\sum_{c=1}^{C_s} x_{m,s,c}}{C_s}\right|\right) \quad \forall s = 1 \dots S, c = 1 \dots C_s$$

- The break time between exams of student is minimized.

$$\min\left(\sum_{m=1}^{M} pod(m)\right)$$

The function $pod(m)$ used to calculate the break time between each exam of student m^{th}. It is designed depending on how each training institution defines the time units.

- Minimize number of courses:

$$\min(C_s) \; \forall s = 1 \dots S$$

Subjects to:

- All students are able to exam

$$x_{m,s,c} * a_{m,s} \geq 1 \; \forall m = 1 \dots m, s = 1 \dots S, c = 1 \dots C_s$$

- A same room, cannot be occupied by two or more examination room at a similar time-slot

$$\sum_{s=1}^{S} \sum_{c}^{C_s} y_{c,s,t,r} \leq 1 \quad \forall t = 1 \dots \mathrm{T}, r = 1 \dots R$$

- One student cannot take two or more exams simultaneously

$$\sum_{m=1}^{M} \sum_{s=1}^{S} \sum_{c=1}^{C_s} \sum_{r=1}^{R} y_{c,s,t,r} * x_{m,s,c} \leq 1 \quad \forall t = 1 \dots T$$

- The number of students in the same examination must be restricted.

$$\Omega \leq \sum_{m=1}^{M} x_{m,s,c} \leq \beta \quad \forall s = 1 \dots S, c = 1 \dots C_s \;\; \forall\, m = 1 \dots M, t = 1 \dots T$$

- Students are not arranged to examination zoom of subject they do not register for

$$\sum_{s}^{S} \sum_{c}^{C_s} x_{m,s,c} \leq \min\left(a_{m,s}, 1\right) \; \forall\, m = 1 \dots M$$

- Course only assigned to the room that satisfy the required conditions.

$$y_{c,s,t,r} * u_{r,f} \geq \; l_{s,f} \quad \forall\, s = 1 \dots S, c = 1 \dots C_s, \; t = 1 \dots T, r = 1 \dots R, f = 1 \dots F$$

- An examination cannot happen at two different shifts.

$$hod(s,c) = 1 \quad \forall\, s = 1 \dots S, c = 1 \dots C_s,$$

hod(s,c) is return the condition of examination $c^{th}$of subject s^{th} whether if it occupied the time of two different days. 1 if it only occupies time in a day. 0 otherwise.

2.2 Compromise programming for proposed MOP

The proposed model optimizes multi objectives, which make it multi-objective programming (MOP). According to Hwang [21], there two frequent used approach to the solution of MOP: preference and non-preference. The preference method takes the solution that best fit to the preference base on the decision-maker. The second approach using a nature compromise solution to all objectives instead of the decision maker. Ngo et al [2][22] used compromise programming to their timetabling and team selection problems. In this study, we want to archived a solution that satisfy all the objectives equally, therefore, we chose to using the second method, the Compromise programming to transform the MOP into single objective problem. The ideal point denoted as $E = \{e_i | i = 1 \ldots 3\}$ and the solution $O = \{o_i | i = 1 \ldots 3\}$ where:

$$e_i = \begin{cases} 0 & if\ i = 1 \\ 0 & if\ i = 2 \\ 0 & otherwise \end{cases}$$

$$o_i = \begin{cases} \sum_{s=1}^{S} fuzzy(\sum_{c=1}^{C_s} \left(\left| \sum_{m=1}^{M} x_{m,s,c} - \frac{\sum_{m=1}^{M} \sum_{c=1}^{C_s} x_{m,s,c}}{C_s} \right| \right)) & if\ i = 1 \\ \sum_{m=1}^{M} fuzzy(pod(X_m, Y_m)) & if\ i = 2 \\ \sum_{s=1}^{S} fuzzy(C_s) & otherwise \end{cases}$$

Where fuzzy function that calculate the score of the parameters on a given scale

Objective function becomes: $min\left(\sqrt{\sum_{i=1}^{3} w_i * norm_i(e_i - o_i)^2}\right)$

Where $norm_i$ represents a normalization function that rearranges the values of the dimension i^{th} in the distance function to a particular range. w_i stands for the weighted parameters of the dimension i^{th} in the fitness function.

3 Genetic Algorithm

The genetic algorithm is a population-based metaheuristic and is a band of Evolution algorithm. This algorithm and its hybrid are usually be used to find the solution to the timetabling problem. The algorithm's idea is to find the best individuals that fit the problem's goal through the hybridization of the best genes from the parent. The hybrid genetic algorithm with greedy initialization proposed by [23] inspire us. To generate the initial Population, we use the greedy algorithm to create a feasible population, where each individual in the population does not violate the model's constraints. We continue to follow the steps of genetic algorithms until the solution is converged.

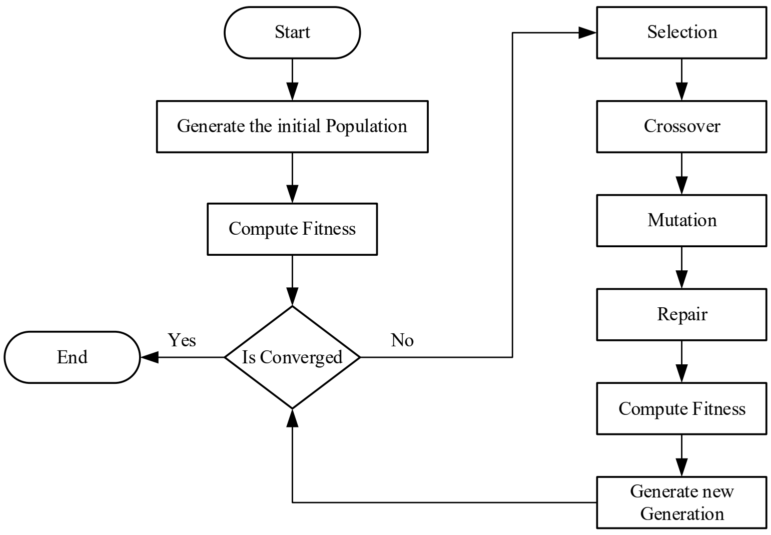

Fig. 1. The flow of proposed genetic algorithm

3.1 Genetic representation and fitness function design

The chromosome represented as follows:

- The chromosome is a vector of S dimensions, the dimension i^{th} represents subject i^{th}.
- For each subject i^{th}, contains the time when the exam of subject i^{th} is hold, and a list of examination rooms that exam subject i^{th}.
- In each examination room i^{th}, there are the room where the examination is hold, and the list of students in the room.

Denote $MObj = MObj_1, MObj_2 \dots, MObj_3\}$ Where:

$$MObj_i = \begin{cases} \beth * S & if\ i = 1 \\ \beth * M & if\ i = 2 \\ \beth * S & otherwise \end{cases}$$

Where $\beth$ is the maximum point of the given scale

The fitness values $P_i^{(g)}.fitness$ could be bias due to the difference in range of its dimensions. We normalize features to the range [0,1]. The Fitness function of the individual can be defined as:

$$P_i^{(g)}.fitness = \sqrt{\sum_{i=1}^{(3)} w_i * \left(e_i - \frac{o_i}{MObj_i}\right)^2}$$

3.2 Genetic operations

Denote:

- U represent the size of population
- $P^e = \{p_i^e \mid i = 1..U\}$ as the population at generation e^{th}
- p_i^e as the individual ith of the population at generation e^{th}
- A_i represents the number of students that need to take subject i^{th}.
- Denote $H_j^{(i,e)}$ as subject j^{th} of individual i^{th} at generation e^{th}
- $H_{j,b}^{(i,e)} = \{0,1\}$. Where $H_{j,b}^{(i,e)} = 1$ represents subject j^{th} is begin at slot b^{th}. 0 otherwise
- Denote $G_c^{(j,i,e)}$ as the examination room c^{th} of subject j^{th} of individual i^{th} at generation e^{th}
- $G_{c,r}^{(j,i,e)} = \{0,1\}$ where $G_{c,r}^{(j,i,e)} = 1$ represents that examination c^{th} is hold in room r^{th} . 0 otherwise.
- w is rate of elites of the population.
- φ is selection rate.
- is the exchange rate of genes between $P_i^{(e)}$ and $P_j^{(e)}$
- Ω as the mutation rate.

The algorithm contains 6 steps:

```
1. Initialize the first population:
   1.1. Randomly generate $P_i^{(e)}$ $\forall i = 1 \dots U$
   1.2. For each $s = 1 \dots S$ in random order:
      1.2.1. t = newPosition($H_s^{(i,e)}$)
      1.2.2. Where function $newPosition(s)$ return a slot that every students have
             to exam subject $s^{th}$ can take, and have enough number of room with
             the same type as subject that can contain every students took the sub-
             ject $s^{th}$.
      1.2.3. $H_{s,t}^{(i,e)}$ =1. $Z_s = t$
      1.2.4. n = $randomCourse(s)$. Where $randomCourse(s)$ return a random
             number between $\sum_{m=1}^{M} A_{m,s} / \partial$ (smallest courses can be created for
             subject $s^{th}$.) and $\sum_{m=1}^{M} A_{m,s} / J_s$(biggest number of courses can be cre-
             ate for subject $s^{th}$)
      1.2.5. $C_s = n$
      1.2.6. $R = choseRoom(n,s)$ Where $choseRoom(n,s)$ return n rooms
             from a set of room $r^{th}$, which $U_{r,f} = 1 \wedge L_{s,f}$=1 and not been used by
             other course at slot $t^{th}$, where $H_{s,t}^{(i,e)}$ = 1. And those n rooms are being
             chosen by descending order from the set of available room.
      1.2.7. $G_{c,r}^{(s,i,e)}$ = 1. $\forall r \in R$ (for each room in R, create an examination corre-
             sponding)
      1.2.8. $Y_{c,s,t',r} = 1. \forall t' = t \dots t + P_s$
```

```
            1.2.9. Divide students of subject $s^{th}$ into each course $\in G^{(j,i,e)}$ equality,
                   where each course has more than $\partial$ students, and less than $J_s$ and
                   $I_r$ where $G_{c,r}^{(s,i,e)} = 1$: $X_{m,s,c}$. $c \in G^{(j,i,e)}$; $A_{m,s} = 1$
2. Selection: Randomly select φ individuals from $P^e$ and return the best individual
   base on fitness value among them.
3. Crossover: Denote $p_{father}^{e}$ and $p_{mother}^{e}$ are parents to crossover:
    3.1. Rearrange $P^{(e)}$ in ascending $P_i^{(e)}.fitness$ order.
    3.2. Declare $Elite$ as the set contains top $U * w$ items of $P^{(e)}$. Randomly
         choose $P_{father}^{(e)}$, $P_{mother}^{(e)}$ from $Elite$.
    3.3. Update $P_{father}^{(e)}$, $P_{mother}^{(e)}$ by a rate, denoted as $ran$. Where: $ran =$
         $rand([0,1])$ return a random probability.
        3.3.1. If $(ran \leq ω)$ Randomly exchange a gen of $P_{father}^{(e)}$ and $P_{mother}^{(e)}$ to
               each other to create two new individuals for the next generation.
               Where exchange a gen is means swap the begin slot, the course list,
               the schedule represents each student and theirs course of this subject to
               each other.
        3.3.2. If $(ω < ran \leq Ω)$ perform mutation on $P_{father}^{(e)}$ or $P_{mother}^{(e)}$ (Described
               in step 5) to create two new individuals for the next generation.
        3.3.3. Otherwise consider $P_{father}^{(e)}$ and $P_{mother}^{(e)}$ 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 takes an individual £ as an input. It contains 3 stage:
    4.1. For each student $m^{th}$ have a subject $s^{th}$ need to be exam but have not been
         assigned to a course yet, find a course that exam subject $s^{th}$ and not full to
         put student $m^{th}$ in.
    4.2. For each room $r^{th}$, that be used by two or more exams simultaneously,
         move the redundant exams to other room of same type, not occupy by other
         exams, and have the capacity more fit to its number of students than the cur-
         rent and change to the found room (if found).
    4.3. For each exam $c^{th}$ that have number of students $< \partial$, fill the course $c^{th}$ by
         other student from other course of the same subject
5. Mutate: Takes an individual £ as an input.
    5.1. Modify position, the courses, and the student assignment randomly of a se-
         lected subject in £.
    5.2. Repair £
6. Repeat 2, 3, and 4 ,5 and set $g = g + 1$ until $b^g = b^{g-1} = \cdots = b^{g-G}$.
```

4 Genetic Algorithm

4.1 Experimental design

To conduct experiments to evaluate the proposed model and algorithm, we collected data of more than 2486 students in the IT department at FPT University, consisting of 74 subjects. We collected more than 9000 exams from those 2500 students mentioned and sort those exams into 100 rooms in 5 days. Each subject has a different requirement maximum number of students. For example, most exams only allow a maximum of 40 students per class, but HCM201 and MLN101 allow 20 more students in their exams. And each day we use 8 hours, 4 hours each shift, provide with 100 rooms with different type, and room's capacity.

We use function $pod(m)$ to calculate the break time between each exam of studentm^{th}. Function $od(m)$ return the total number of slots between each exam of student m^{th}, two exams happen on different days, and the return value has to add several slots that equal the number of a slot of a shift for each day apart.

$$Q_s = A_{m,s} * Z_s . \forall s = 1 \dots S$$

$$E, F = sort(Q)$$

$$pod(m) = \sum_{i=1}^{\sum_{s=1}^{S} A_{m,s} - 1} E_i - E_{i+1} + P_{F_{i+1}}$$

Where function $sort(Q)$ return array E is the descending order of array Q, and array F is the index that correspond to array E.

We also use function $hod(s, c)$ to check if the slot the course is held is invalid (between 2 shifts) or not, which is denoted as:

- n as the number of slots per day.
- Number of slots per shift denoted as m.
- α represents the slot that the exam begins.
- β stands for the slot that the exam begins.
- The invalid shift denoted as: $\alpha < \lfloor T/n \rfloor * n + m < \beta$
- The invalid day presented as: $\alpha < \lfloor T/n \rfloor * n + n < \beta$

Because in the experience, we notice the return value of each objective are in difference scale. It is because while, objective balance the number of students in exams of a subject, and minimize the cost of resource, are easy to archive. However, objective minimize the break-time between of exams are harder to minimize, because the diversity in the type of exams of each student. This phenomenon, may make some objective such as objective insignificant in the total value. Therefore, we decide to use fuzzy method [29] to balance the scale of each objective. In the experience we fuzzy the value of each objective by using function $fuzzy\ i^{th}$ for the respective objective. Where:

$$fuzzy1 = \frac{\sum_{s=1}^{S} obj1(s)}{S*10}$$

Where $obj1(s) =$

$$\begin{cases} \frac{\sum_{c=1}^{C_s}\left(\left|\sum_{m=1}^{M} x_{m,s,c} - \frac{\sum_{m=1}^{M}\sum_{c=1}^{C_s} x_{m,s,c}}{C_s}\right|\right)}{10} & if \ \sum_{c=1}^{C_s}\left(\left|\sum_{m=1}^{M} x_{m,s,c} - \frac{\sum_{m=1}^{M}\sum_{c=1}^{C_s} x_{m,s,c}}{C_s}\right|\right) \le \sum_{c=1}^{C_s} 10 \\ 10 \ otherwise & \end{cases}$$

$$fuzzy2 = \frac{\sum_{m=1}^{M} obj2(m)}{M*10}$$

$$obj2(m) = \begin{cases} 0 \ \ if \ pod(X_m, Y_m) = 0 \\ 1 \ \ if \ pod(X_m, Y_m) > 0 \land pod(X_m, Y_m) \le \frac{\mathrm{T} - \sum_{s=1}^{S} A_{m,s} * P_s}{10} \\ 2 \ \ if \ pod(X_m, Y_m) > \frac{\mathrm{T} - \sum_{s=1}^{S} A_{m,s} * P_s}{10} \land pod(X_m, Y_m) \le 2 * \frac{\mathrm{T} - \sum_{s=1}^{S} A_{m,s} * P_s}{10} \\ 3 \ \ if \ pod(X_m, Y_m) > 2 * \frac{\mathrm{T} - \sum_{s=1}^{S} A_{m,s} * P_s}{10} \land pod(X_m, Y_m) \le 3 * \frac{\mathrm{T} - \sum_{s=1}^{S} A_{m,s} * P_s}{10} \\ 4 \ \ if \ pod(X_m, Y_m) > 3 * \frac{\mathrm{T} - \sum_{s=1}^{S} A_{m,s} * P_s}{10} \land pod(X_m, Y_m) \le 4 * \frac{\mathrm{T} - \sum_{s=1}^{S} A_{m,s} * P_s}{10} \\ 5 \ \ if \ pod(X_m, Y_m) > 4 * \frac{\mathrm{T} - \sum_{s=1}^{S} A_{m,s} * P_s}{10} \land pod(X_m, Y_m) \le 5 * \frac{\mathrm{T} - \sum_{s=1}^{S} A_{m,s} * P_s}{10} \\ 6 \ \ if \ pod(X_m, Y_m) > 5 * \frac{\mathrm{T} - \sum_{s=1}^{S} A_{m,s} * P_s}{10} \land pod(X_m, Y_m) \le 6 * \frac{\mathrm{T} - \sum_{s=1}^{S} A_{m,s} * P_s}{10} \\ 7 \ \ if \ pod(X_m, Y_m) > 6 * \frac{\mathrm{T} - \sum_{s=1}^{S} A_{m,s} * P_s}{10} \land pod(X_m, Y_m) \le 7 * \frac{\mathrm{T} - \sum_{s=1}^{S} A_{m,s} * P_s}{10} \\ 8 \ \ if \ pod(X_m, Y_m) > 7 * \frac{\mathrm{T} - \sum_{s=1}^{S} A_{m,s} * P_s}{10} \land pod(X_m, Y_m) \le 8 * \frac{\mathrm{T} - \sum_{s=1}^{S} A_{m,s} * P_s}{10} \\ 9 \ \ if \ pod(X_m, Y_m) > 8 * \frac{\mathrm{T} - \sum_{s=1}^{S} A_{m,s} * P_s}{10} \land pod(X_m, Y_m) \le 9 * \frac{\mathrm{T} - \sum_{s=1}^{S} A_{m,s} * P_s}{10} \\ 10 \ \ otherwise \end{cases}$$

$$fuzzy1 = \frac{\sum_{s=1}^{S} obj3(s)}{S*10}$$

$$obj3(s) = \begin{cases} C_s - \left\lceil \frac{\sum_{m=1}^{M} A_{m,s}}{\partial} \right\rceil & if \ C_s - \left\lceil \frac{\sum_{m=1}^{M} A_{m,s}}{\partial} \right\rceil < 10 \\ 10 \quad otherwise & \end{cases}$$

This experience is implemented in java 1.8.0 and executed on the computer with detail configuration as CPU Intel core i5-8300H @2.30Hz, 8GB. We evaluated both of the proposed algorithms on the tested dataset.

4.2 Results

The GA are governed by a series of parameters. We execute the algorithms several times with collected dataset to indicate the best set of parameters and weights that show in Table 1.

Table 1. Parameters and weights to execute the algorithms

Parameter	Value
Number of populations β	0.3*S
Exchange genes rate μ	0.6
Mutation rate ϖ	0.3
Selection rate υ	0.5
Number of elites Ω	0.1

Meta-heuristic algorithms do not guarantee to find the optimal solution with different initialization values. So, we run the algorithms 10 times with different initializations to find the schedule for 2500 students to exam 74 subjects. Figure 2 shows the result of 10 executions with the data of 2500 students. It can be observed from Figure 2.A that the difference of fitness value in each execution are not too significant, which show the stability of the algorithm. It can be observed that the flow of fitness values and F2 values are similar to each other. Since the return value of objective 2 are double the F1 and F3 value, it also influences to the flow to the fitness value. The reason for this phenomenon is because the data for the experience is an enrolment-base dataset, therefore each student in a same exam have to take many different other exams. So, in order to minimize all the break-time between exams of all student is hard to archive. Which make the minority students have to satisfy their time for the majority, and prevent F2 to decrease to a certain point.

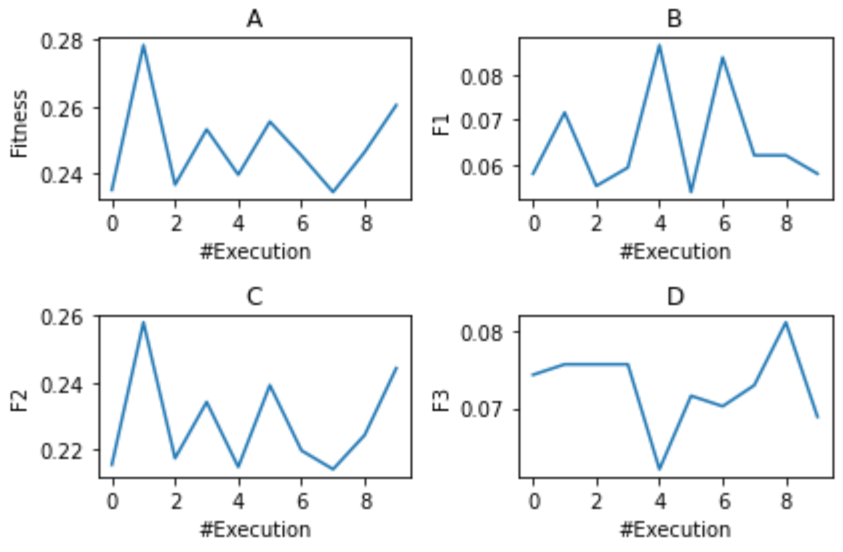

Fig. 2. Outputs over 10 executions. A, B, C, D are the value of Fitness, F1, F2, F3 respectively

Figure 3 illustrate the ability to convert of the algorithm over generations. We stop the algorithm at generation 350. It can be seen that the algorithm archives its best solution at generation 300. The convergence rate of the algorithm shows that population diversity can be preserved. Usually, schemes that make the algorithm converge early often lose diversity. Making it difficult for the next generations to make a good mutation.

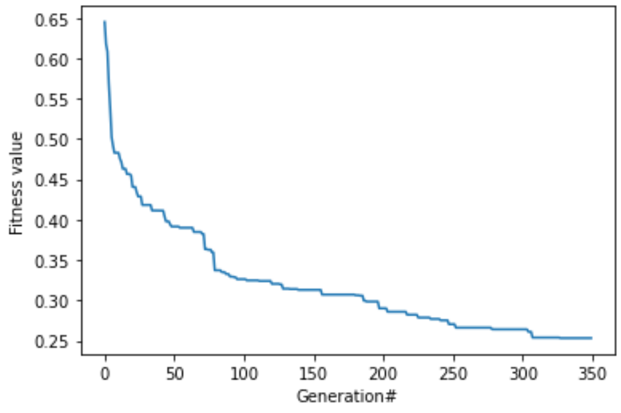

Fig. 3. Fitness values over generations

Figure 4, 5, 6 illustrate some detail data of the solution with the best fitness value. In Figure 4, we category students base on the number of days that they have to exam. It can be observed that most of students only need 1 to 3 days to finish all of their exams. While there are still some students' schedule require 4 days to finish, and insignificant number of students require 1 to 2 extra days to finish the examination. The reason for some students requires 5 to 6 days to finish their exams.

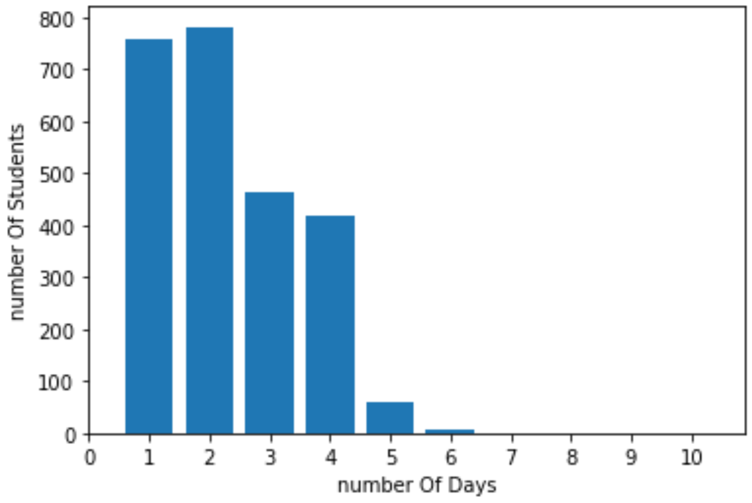

Fig. 4. Category students by the number of days they have to exam.

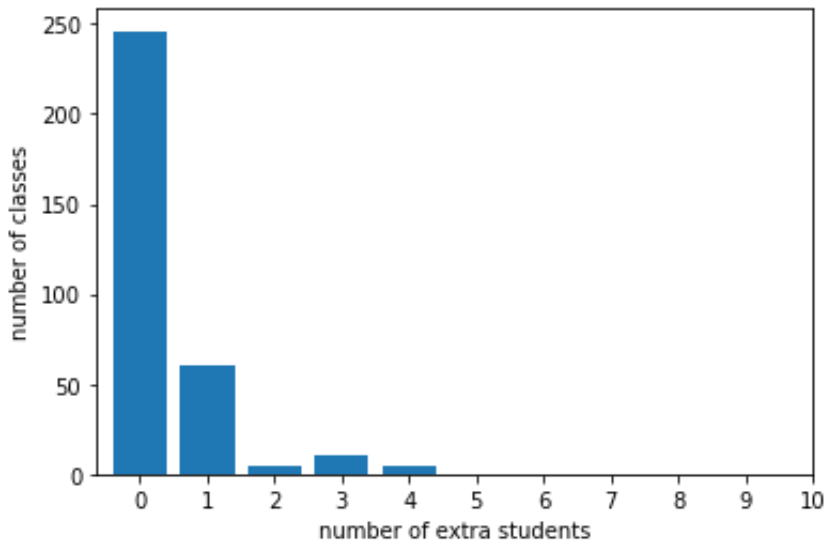

Fig. 5. Category exams by the different of students to the average

The changed values of Objective 1 are illustrated in Figure 5. The chart group exams according to the students' difference in each exam to the average number of students in the respective subject. Most of the exams are equals or have only one student difference. There are still exist some exams that have more considerable differences, the maximum is 6, but the number is insignificant. The case of 5 or 6 students is different between classes, usually because the solution decides to use rooms with various capacities to reduce resources' cost. Therefore, the return value of objective 1 is smaller compare to other objectives.

Figure 6 illustrates the number of exams is created compared to the minimum requirement in each subject. According to the formation, the differences are not too significant in the solution compared to the ideal number. Most of the subjects that require more exams are the ones which have a larger number of students compare to other subjects. Those subjects need more exams to avoid using rooms with different capacities. While in the solution, each objective has some aspects that differ from the ideal result. However, those issues exist because of the balance of each objective and the scales and data design.

We can see that the second objective is more challenging to optimize than the other objectives. The data in Figure 4 also show some insignificant cases that the break-time is not optimized, which is a small number of students still need 4 to 5 days to finish their exams. The cause of this phenomenon is because the experiences are enrollment-based, in which students do not follow a similar curriculum with each other but take the courses and exams independently from others. The diversity of different subjects in a subject is illustrated in Figure 7. According to the figure, some subjects have students take around 50 other subjects. Which makes it is hard to satisfy the requirement to minimize the break time for all students in a subject.

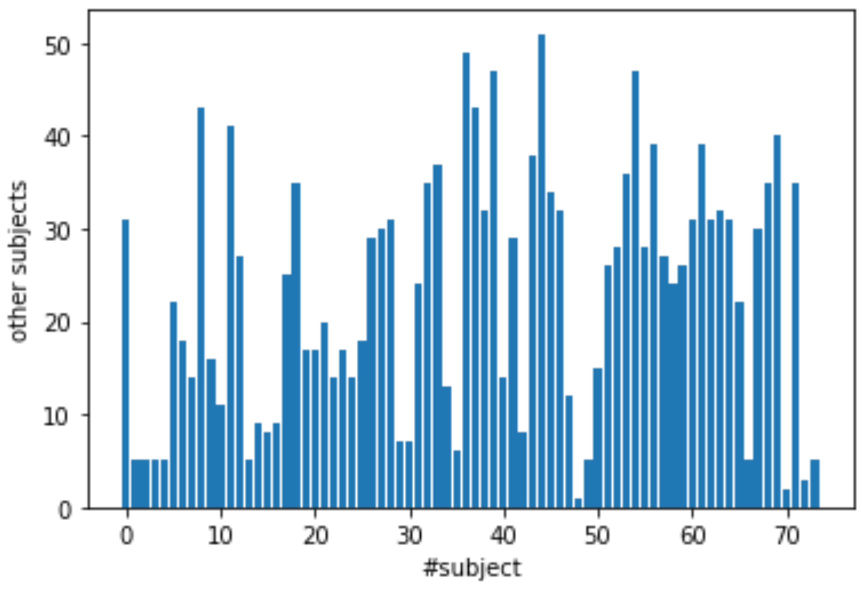

Fig. 6. The diversity of subjects

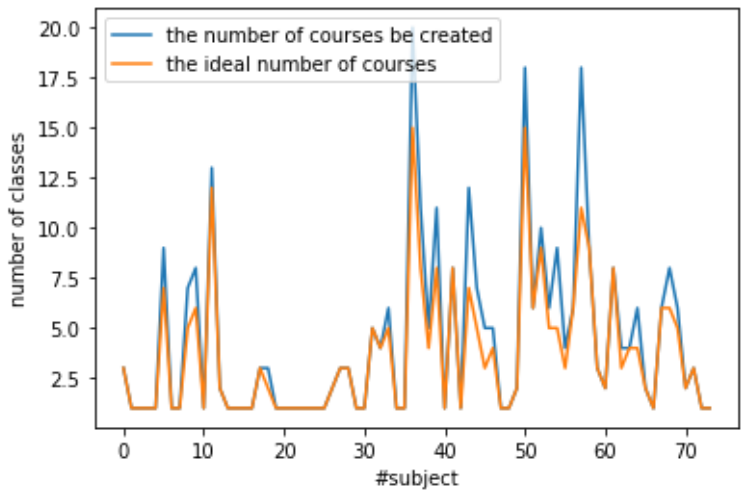

Fig. 7. Number of created exams compare to ideal number of exams

By changing the dimensions' weight parameter, we can customize the result of the algorithm. Figure 8 shows different sets of products with varying sets of parameters. In those executions, since we use smaller data, the students are divided into groups and have to take the same exams with others in the same group. Therefore, it possible for the solution to archive the ideal result when change the weight parameters focus on a single objective. That will minimize each objective's influence on each other and result that the best solution for the focused objective while satisfying other objectives. We can observe the impact between the number of exams per subject and students' balance in exams of the same subject in the first and third execution. With each objective archive the ideal result in the respective executions, the other value is increased significantly. On the other hand, while the relationship of the break-time objectives with others is not healthy as resource minimize objectives and balance student, but to archive the ideal solution, it still requires other objectives have to satisfy their condition.

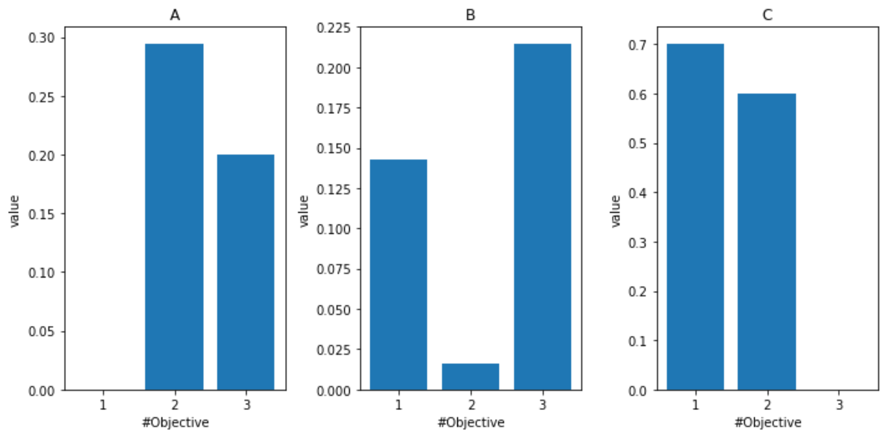

Fig. 8. Result after 3 executions with different weight parameters. A) $w_1 = 1000, w_2 = 1, w_3 = 1$. B) $w_1 = 1, w_2 = 1000, w_3 = 1$. B) $w_1 = 1000, w_2 = 1, w_3 = 1000$.

We visualize the feasible optimal solution of the algorithm in each pair of objectives. The Pareto Frontier was obtained by executing the algorithm 250 times with different weights parameters shown in Figure 9.

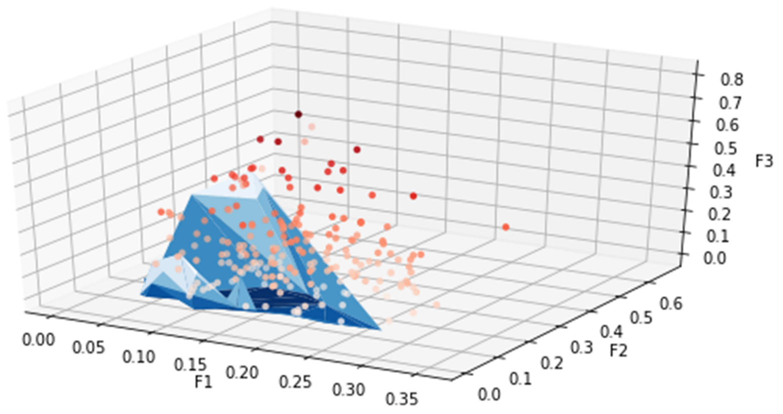

Fig. 9. . Obtained Pareto Frontier from 250 executives with different weight parameter

5 Conclusion

In this study, we have designed a new approach to examination timetabling. Comparing with previous studies, our proposed model is MOP. It has access to many business aspects, primarily of interest to students than any previous model. which seeks typically to save costs. Not only proposing the optimal model, but we also use a hybrid approach for the MOP problem. It is a combination of an idea point and linear scalarizing. This method allows even if decision-makers cannot assign preferences for each specific factor of the problem. The model still directs the algorithm towards an idea point. We also design a scheme of the Genetic Algorithm to solve a proposed optimal model. The results showed that the algorithm-maintained population diversity. However, the way we transform the objective values to the same scale is not good enough. It is not practical to balance the importance of the objective functions. It requires some groping to indicate the values for the weight parameters of the target functions. Once these parameters are defined, the program is an efficient tool that allows decision-makers to manipulate agents. In the future, we continue to improve the algorithm to increase population diversity [24] and improve scaling for the target functions to balance their importance. Improving computing performance with parallel computing is also one of our priorities [25], [26].

6 References

[1] Babaei, Hamed; Karimpour, Jaber; Hadidi, Amin (2015). A survey of approaches for university course timetabling problem. Computers & Industrial Engineering, 86(), 43–59. https://doi.org/10.1016/j.cie.2014.11.010

[2] Son, Ngo T.; Jaafar, Jafreezal; Aziz, Izzatdin A.; Anh, Bui N. 2021. "A Compromise Programming for Multi-Objective Task Assignment Problem" Computers 10, no. 2: 15. https://doi.org/10.3390/computers10020015

[3] ALJARAH, Ibrahim; SALHIEH, Ayiad; FARIS, Hossam. An Automatic Course Scheduling Approach Using Instructors' Preferences. International Journal of Emerging Technologies in Learning (iJET), [S.l.], v. 7, n. 1, p. pp. 24-32, Feb. 2012. ISSN 1863-0383. Available at: <https://www.online-journals.org/index.php/i-jet/article/view/1881/2140>. Date accessed: 31 Jan. 2021. https://doi.org/10.3991/ijet.v7i1.1881

[4] R. Qu; E. K. Burke; B. McCollum; L. T. G. Merlot; S. Y. Lee (2009). A survey of search methodologies and automated system development for examination timetabling., 12(1), 55–89. https://doi.org/10.1007/s10951-008-0077-5

[5] Cooper, T.B., Kingston, J.H.: The complexity of timetable construction problems. In: Proc. of the 1st Int. Conference on the Practice and Theory of Automated Timetabling (ICPTAT '95). (1995) 511–522

[6] Liyana, Nurul & NAH, Aizam. (2017). University course timetabling and the requirements: Survey in several universities in the east-coast of Malaysia. AIP Conference Proceedings. 1870. 040013. 10.1063/1.4995845. https://doi.org/10.1063/1.4995845

[7] Acostamado, R.J. & Villa Marulanda, Marcela. (2013). An Integer Programming Model for The Academic Timetabling Problem. 10.13140/2.1.3910.9122.

[8] Mccollum, Barry & McMullan, Paul & Parkes, Andrew & Burke, Edmund & Qu, Rong. (2012). A new model for automated examination timetabling. Annals OR. 194. 291-315. 10.1007/s10479-011-0997-x. https://doi.org/10.1007/s10479-011-0997-x

[9] DENER, Murat & CALP, M. Hanefi. (2018). Solving the Exam Scheduling Problems in Central Exams With Genetic Algorithms. Mugla Journal of Science and Technology. 4. 102-115. 10.22531/muglajsci.423185. https://doi.org/10.22531/muglajsci.423185

[10] Özcan, Ender & Ersoy, E. (2005). Final exam scheduler - FES. 2005 IEEE Congress on Evolutionary Computation, IEEE CEC 2005. Proceedings. 2. 1356 - 1363 Vol. 2. 10.1109/CEC.2005.1554848. https://doi.org/10.1109/cec.2005.1554848

[11] Shatnawi, Ali & Al-Qahtani, Hadi & Fraiwan, Mohammad. (2017). Exam Scheduling: A Case Study. 10.1109/ICACI.2017.7974498. https://doi.org/10.1109/icaci.2017.7974498

[12] Erdős, Szilvia & Kovari, Bence. (2019). Genetic Algorithm Based Solution for Final Exam Scheduling. 10.26649/musci.2019.021. https://doi.org/10.26649/musci.2019.021

[13] Yang, Xiaofei & Ayob, Masri & Ahmad Nazri, Mohd Zakree. (2017). An investigation of timetable satisfaction factors for a practical university course timetabling problem. 1-5. 10.1109/ICEEI.2017.8312409. https://doi.org/10.1109/iceei.2017.8312409

[14] He yan, song-nian yu, a multiple neighborhoods-based simulated annealing algorithm for timetable problem, springer-verlag berlin heidelberg, gcc 2003, pp. 474-481 2004

[15] Ayob, Masri & Jaradat, Ghaith. (2009). Hybrid Ant Colony Systems for course timetabling problems. 2009 2nd Conference on Data Mining and Optimization, DMO 2009. 120 - 126. 10.1109/DMO.2009.5341898. https://doi.org/10.1109/dmo.2009.5341898

[16] Mandal, Ashis & Mohmad Kahar, Mohd Nizam. (2015). Solving examination timetabling problem using partial exam assignment with great deluge algorithm. I4CT 2015 - 2015 2nd International Conference on Computer, Communications, and Control Technology, Art Proceeding. 530-534. 10.1109/I4CT.2015.7219635. https://doi.org/10.1109/i4ct.2015.7219635

[17] Y. Yang, W. Gao and Y. Gao, "Mathematical modeling and system design of timetabling problem based on improved GA," 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Guilin, 2017, pp. 214-220, https://doi.org/10.1109/fskd.2017.8393102

[18] Aldeeb, Bashar & Md Norwawi, Norita & Al-Betar, Mohammed & Jali, Zalisham. (2015). Solving University Examination Timetabling Problem Using Intelligent Water Drops Algorithm. 187-200. 10.1007/978-3-319-20294-5_17. https://doi.org/10.1007/978-3-319-20294-5_17

[19] Badoni, Rakesh P.; Gupta, D.K. (2014). A graph edge colouring approach for school timetabling problems. International Journal of Mathematics in Operational Research, 6(1), 123–. https://doi.org/10.1504/ijmor.2014.057853

[20] WEN-JING, Wang. Improved Adaptive Genetic Algorithm for Course Scheduling in Colleges and Universities. International Journal of Emerging Technologies in Learning (iJET), [S.l.], v. 13, n. 06, p. pp. 29-42, may. 2018. ISSN 1863-0383. Available at: <https://www.online-journals.org/index.php/i-jet/article/view/8442>. Date accessed: 01 Feb. 2021. https://doi.org/10.3991/ijet.v13i06.8442

[21] Ching-Lai Hwang; Abu Syed Md Masud (1979). Multiple objective decision making, methods and applications: a state-of-the-art survey. Springer-Verlag. ISBN 978-0-387-09111-2. https://doi.org/10.1002/zamm.19800600932

[22] Ngo, T.S.; Bui, N.A.; Tran, T.T.; Le, P.C.; Bui, D.C.; Nguyen, T.D.; Phan, L.D.; Kieu, Q.T.; Nguyen, B.S.; Tran, S.N. Some Algorithms to Solve a Bi-Objectives Problem for Team Selection. Appl. Sci. 2020, 10, 2700. https://doi.org/10.3390/app10082700

[23] Ilyas, Rafia & Iqbal, Zahid. (2015). Study of Hybrid Approaches used for University Course Timetable Problem (UCTP). https://doi.org/10.1109/iciea.2015.7334198

[24] Cheng Chen, Zhenyu Yang, Yuejin Tan, Renjie He, "Diversity Controlling Genetic Algorithm for Order Acceptance and Scheduling Problem", Mathematical Problems in Engineering, vol. 2014, Article ID 367152, 11 pages, 2014. https://doi.org/10.1155/2014/367152

[25] WU, Liping. The application of Coarse-Grained Parallel Genetic Algorithm with Hadoop in University Intelligent Course-Timetabling System. International Journal of Emerging Technologies in Learning (iJET), [S.l.], v. 10, n. 8, p. pp. 11-15, dec. 2015. ISSN 1863-0383. Available at: <https://www.online-journals.org/index.php/i-jet/article/view/5206>. Date accessed: 31 Jan. 2021. https://doi.org/10.3991/ijet.v10i8.5206

[26] N. T. Son, J. Jaafar, I. A. Aziz and B. N. Anh, "Meta-Heuristic Algorithms for Learning Path Recommender at MOOC," in IEEE Access, vol. 9, pp. 59093-59107, 2021 https://doi.org/10.1109/access.2021.3072222

7 Authors

Ngo Tung Son graduated bachelor's degree in Computing in 2011, and master degree in Computer Science since 2014 from the Vietnam-France University, Vietnam. He is currently pursuing his Ph. D. in Information Technology at Universiti Teknologi PETRONAS (UTP), Malaysia. From 2014 to 2016, he worked at the Panasonic R&D center in Hanoi, Vietnam as a software engineer. From early 2016, he worked at 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 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, UTP, in 1999, where he was the Former Head, from 2012 to 2016. He is currently an Associate Professor and the Dean of the Faculty of Science and Information Technology, Universiti Teknologi PETRONAS (UTP), Malaysia. He is also the Head of the Center for Research in Data Science (CERDAS), which focus on implementation of AI and machine learning in oil and gas (O and G) industry. He is also active in conducting professional training in AI and big data analytic for 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 had become a member of the Academy Science Malaysia SIG in Machine Learning, 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 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), where he focuses in solving complex upstream oil and gas (O&G) industry problems from the viewpoint of computer sciences and data analytics. He has secured and delivered numerous computing-related O&G projects mainly relating to seismic data processing and implementation of workflows for processing hydrocarbon exploration dataset on cloud-based systems. He is also actively working on real industrial projects to predicting machine failure at oil and gas platforms and refineries through machine learning approaches in which most of his research are mainly funded by the Petroleum Research Fund, PETRONAS, Yayasan UTP fund, and The Malaysian Ministry of Higher Education

Nguyen Hoang Giang is currently studying IT engineer degree at FPT University. His major is Software Engineering. Nguyen is also a research assistant at SAP-LAP FPT University laboratory. His main job is to develop mobile and web applications. Nguyen research interests include artificial intelligence, adaptive and intelligent systems, decision support systems, data mining and robotic.

Bui Ngoc Anh graduated with an engineer degree and a master's degree in computer science from Hanoi University of Technology in 2002 and 2006. Since 2003 he has worked on various projects in the software industry. With nearly 20 years of experience in the software industry, Bui is the one of the experts at FPT Technology Corporation in Vietnam. In 2011, Bui started his work as an IT lecturer at FPT University, Vietnam. Since 2019, Bui is Head of Computing Fundamentals Department at FPT University. He is also the leader of SAP-LAP FPT Laboratory at FPT University. His research interests include big data, data mining, computer vision and software engineering.

Article submitted 2021-01-10. Resubmitted 2021-02-01. Final acceptance 2021-02-02. Final version published as submitted by the authors.