




Article

A Compromise Programming for Multi-Objective Task Assignment Problem

Son Tung Ngo ^{1,2,*} , Jafreezal Jaafar ¹, Izzatdin Abdul Aziz ¹  and Bui Ngoc Anh ² 

¹ Center for Research in Data Science, Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Tronoh 32610, Malaysia; jafreez@utp.edu.my (J.J.); izzatdin@utp.edu.my (I.A.A.)

² Information and Communication Technology Department, FPT University, Hanoi 100000, Vietnam; anhbn5@fe.edu.vn

* Correspondence: sonnt69@fe.edu.vn

Abstract: The problem of scheduling is an area that has attracted a lot of attention from researchers for many years. Its goal is to optimize resources in the system. The lecturer's assigning task is an example of the timetabling problem, a class of scheduling. This study introduces a mathematical model to assign constrained tasks (the time and required skills) to university lecturers. Our model is capable of generating a calendar that maximizes faculty expectations. The formulated problem is in the form of a multi-objective problem that requires the trade-off between two or more conflicting objectives to indicate the optimal solution. We use the compromise programming approach to the multi-objective problem to solve this. We then proposed the new version of the Genetic Algorithm to solve the introduced model. Finally, we tested the model and algorithm with real scheduling data, including 139 sections of 17 subjects to 27 lecturers in 10 timeslots. Finally, a web application supports the decision-maker to visualize and manipulate the obtained results.

Keywords: timetabling; task assignment; MOP; combinatory optimization; compromise programming; genetic algorithm



Citation: Ngo, S.T.; Jaafar, J.; Aziz, I.A.; Anh, B.N. A Compromise Programming for Multi-Objective Task Assignment Problem. *Computers* **2021**, *10*, 15. <https://doi.org/10.3390/computers10020015>

Received: 14 December 2020

Accepted: 20 January 2021

Published: 25 January 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

1.1. Background

In many fields such as production management, engineering . . . etc., scheduling plays a key role in productivity enhancement. The scheduling process is responsible for assigning machines and resources to tasks such that all tasks are executed while meeting business constraints. In the training institutions, the scheduling problems are mostly presented in the form of timetabling problems. The university timetabling problem's goal is to find a method to allocate the predefined resources that minimize the cost where all constraints within the problem must be satisfied. The resources here consist of classes (groups of students with the same schedule), a subject that requires one or more specific skills and knowledge, time slots that determine when a particular class and subject are attached. The university usually performs a scheduling task before a semester begins [1–5]. The teaching assignment problem, in essence, is a branch of the timetabling problem. The scheduler must assign courses to trainers to satisfy the professional constraints and working time [6]. A lecturer can teach only the courses that they are qualified in, and at their available timeslots.

This research was conducted on a practical case study at FPT University in Vietnam. Currently, the university's scheduling process is a manual process. In the situation, the student can register for their studies very soon before the department head has enough resources to determine the final schedule (of course, some classes could be canceled due to lack of resources later). The training department creates groups of students who would like to study the same subject based on the registrations and select the time slots. However, the department heads still need to assign their lecturer to teach these classes later. The reason for this is that we are student-centered. Other resources revolve around students to

support them. In short words, the lecturers' timetables are considered last. The project aims to provide an automated task assignment tool to replace the manual process of matching lecturers to their courses, as shown in Figure 1. The system assigns these courses to the lecturers based on their skills and expectations (more detail in Section 2).

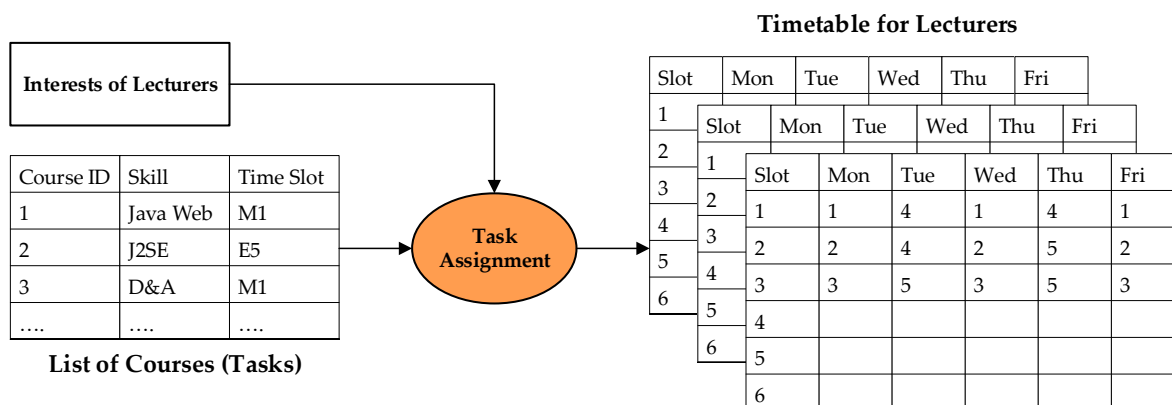


Figure 1. The teacher assignment problem.

1.2. Related Work

There have been many studies on the problem of university scheduling. Many of them have used an integer programming (IP) model to formulate the problem. For example, Andrade et al. have built a Non-Linear Binary Integer Programming mathematical model to develop the school timetabling problem, which is used to assign teaching tasks to teachers at a defined time frame [1]. Gianpaolo et al. proposed an Integer Programming formulation of selecting the training offer and the related timetabling for high-school remedial courses subject to constraints on budget and business operations [3]. Daskalaki et al. presented a binary integer programming model of the university timetabling problem, which tries to minimize the linear cost function [7]. Feng et al. developed a mixed-integer linear program for the university timetabling problem. The original problem converted to the three-dimensional container packing problem. They consider day, period, and room as the three dimensions of one container and the lectures as different sized items then assign them into the container [8].

Several researchers proposed models focused on assignments for rooms and time slots to achieve workable schedules while optimizing the lecturer's interests. For example, Nouri and Driss [9,10] use the multi-agent approach, where the agents represent teachers of different levels and seek to assign their lectures according to their interests. Higher-ranking teachers are given priority in meeting their interests. Malik et al. built a model for mapping the task to the lecturer that maximizes their preference on the time-slot [11]. There are many different views about the compact schedule. The goal is to avoid idle time on the teacher's plan and minimize working days [12,13].

The task assignment problems exist in many different forms. While some of them, like the classical problem, have polynomial-time solutions [14], others are NP-hard combination optimization problems [15] that require approximation approaches. A metaheuristic is widely known [16]. In [17], Lewis classifies several metaheuristic-based techniques into three classes for University Timetabling problems in their survey. Muthuraman and Venkatesan also conducted a survey of meta-heuristic algorithms for solving combinatorial optimization problems [18]. They reviewed several algorithms, such as ant colony optimization, evolutionary computation, particle swarm optimization, etc. Many researchers used genetic algorithms, an evolutionary algorithm to solve scheduling problems, and task assignment problems [19–21]. Genetic Algorithm generates high-quality solutions to optimization and search problems. A particular researcher can have different designs of the Genetic Algorithm to solve specific problems. Feng et al. combine genetic algorithms and search strategies to create offspring in populations based on information collected

from the best individuals of previous generations and with a local search that improves the efficacy of the proposed Genetic Algorithm [8]. Yang develops an efficient hybrid genetic algorithm based on algorithms for the converted problem [22]. In [23], Corne suggested some of timetabling constraints including: unary constraints, binary constraints, capacity constraints, event spread constraints, agent constraints.

The common of the above studies is that they basically have the right target orientation to handle timetabling. However, the proposed model is too simple. It does not cover enough the achievements that complex business required. University lecturers are experts, enabling them to perform their jobs as important as cost optimization. The preferences of the trainer have not been properly considered in the resource optimization process. The use of metaheuristic algorithm to solve the combinatorial optimization problem is a reasonable direction. This study introduces a multi-objective optimization problem that used binary integer decision variables and a version of the Genetic Algorithm to solve the task assignment problem mentioned in Section 1.1.

1.3. Contribution

In this study, we present an approach to construct a task assignment support system for the university—the research output, including an optimization model, algorithm, and software application for actual use. The application plays a stage in the automatic scheduling solution at FPT University. It is a new variant of the teaching task assignment problem. The related research to the scheduling and task assignment may benefit from our study. Our mission is to arrange the teaching tasks for available human resources. We built a multi-objectives model that accesses each individual's level of interest assigned to the job, providing a binding compliance solution. Our proposed model covers more business requirements than previous works since the many aspects of lecturer preferences are considered. The optimization model of the problem is described in Section 2.

There are many ways to solve the proposed optimization model. We choose compromising programming to transform the multi-objective problem into a single-objective problem. Each MOP approach has its advantages and disadvantages and is suitable for different decision-makers groups. Still, compromise programming works extremely better if no preference is indicated (it is assumed that the people are treated equally). We have implemented a Genetic Algorithm that solves both optimal models of a target mentioned above—the detail of the implementation is described in Section 3. In the following sections of this paper, we present our experiments using the scheduling data of Computer Fundamentals at FPT University. A review of the algorithm is also discussed in Section 4. The remaining are discussion and conclusion.

2. Mathematical Problem

2.1. Multi-Objective Task Assignment Problem

In this research, we also define our timetable problem in the form of IP as follows:

- Let G is the number of lecturers.
- Denote S is the number of subjects.
- T is the number of available time slots, in our case, $T = 10$ as described in Table 1.
- H is the number of section, a section represents a particular class studies a specific subject at a timeslot.
- c_h, s_h, t_h are class, subject, and time slot of section h -th respectively.
- D_g is a number of classes that lecturer g -th prefers to teach.
- M_g is a minimum number of classes that the lecturer g -th has to teach.
- $a_{s,g} \geq 0$ as integer for every $s = 1 \dots S, g = 1 \dots G$ represent the rating of the lecturer g -th to teach subject s -th. The value 0 indicates that the lecturer does not want to teach the subject. Other values respectively mean “like a little” to “like very much”.
- $b_{t,g} \geq 0$ as integer for every $t = 1 \dots T, g = 1 \dots G$ denote the rating of the lecturer g -th to teach at time slot t -th. The value 0 indicates that the lecturer does not want to teach at the time slot. Other values respectively mean “like a little” to “like very much.”

- $x_{h,g}$ is the decision variable for every $h = 1 \dots H$, $g = 1 \dots G$. $x_{h,g} = 1$ if the lecturer g -th is assigned to section h -th, $x_{h,g} = 0$ otherwise.

Table 1. The details of 10-time slots defined at the FPT University for a week.

Time-Slot \ DoW	Monday	Tuesday	Wednesday	Thursday	Friday	Part of the Day
1	M1		M1	M4	M1	Morning
2	M2	M4	M2	M5	M2	
3	M3	M5	M3		M3	
4	E1		E1	E4	E1	After Noon
5	E2	E4	E2	E5	E2	
6	E3	E5	E3		E3	

We define several constraints to the problem as follows:

- All section must be assigned lecturer and at most one lecturer is assigned to a section.

$$\sum_{g=1}^G x_{h,g} = 1 \quad \forall h = 1 \dots H \quad (H1)$$

- A particular lecturer does not teach the subject that he/she does not have skill.

$$a_{s_h,g} \geq x_{h,g} \quad \forall h = 1 \dots H, g = 1 \dots G \quad (H2)$$

- A particular lecturer does not teach at the time-slot that he/she is not available.

$$b_{t_h,g} \geq x_{h,g} \quad \forall h = 1 \dots H, g = 1 \dots G \quad (H3)$$

- All lecturers have to satisfy the quota for the number of sections they have to teach.

$$\sum_{h=1}^H x_{h,g} \geq M_g \quad \forall g = 1 \dots G \quad (H4)$$

In this research, we have defined some objectives functions that maximize the lecturer's preference level on time-slots, subjects, and the number of classes that the lecturer expects to teach. The objective functions described as follows:

- Maximize the expectations of the lecturers on the subject they want to teach.

$$\max \left\{ \sum_{h=1}^H x_{h,g} * a_{s_h,g} \right\} \quad \forall g = 1 \dots G \quad (O1)$$

- Maximize the expectations of the lecturers on the time slots they want to teach.

$$\max \left\{ \sum_{h=1}^H x_{h,g} * b_{t_h,g} \right\} \quad \forall g = 1 \dots G \quad (O2)$$

- Minimize the errors on the number of classes that the lecturers want to teach.

$$\min \left\{ \left| \sum_{h=1}^H x_{h,g} - D_g \right| \right\} \quad \forall g = 1 \dots G \quad (O3)$$

Minimize the number of parts of the day, which lecturers must work (morning, afternoon every day). The lecturer would register three classes, even if he expressed his interest in all of the time-slots. It is better to assign him/her to work in the slot-times (E1, E2, E3) instead of (E1, E4, M1):

$$\max \left\{ pod \left(\left\{ x_{h,g} \mid h = 1 \dots H \right\} \right) \right\} \quad \forall g = 1 \dots G \quad (O4)$$

where *pod* is a fuzzy logic membership function that returns the rating for the number of parts of the day, which lecturers have to work, the detailed implementation can be different in different situations. We show our implementation in the part of the experiment to suit the context of FPT University.

The proposed model is in the form of a multi-objective programming problem (MOP) [24]. Since there are often many Pareto optimization solutions for MOP problems, solving such a problem is not as simple as a typical single goal optimization problem. In the following sections, we present an approach to transform the optimal problem into a more suitable form to find the optimal solution in the decision space.

2.2. Compromise Programming for MOP

Our proposed scheduling problem becomes MOP. There are two main approaches to solving the MOP problem: preference method and non-preference method, as mentioned in Hwang's survey [24]. The most useful solution is found using different philosophies that depending on the subjective preferences of the decision-makers. In the decision-making process, decision-makers can place interest in each criterion according to his/her subjective preferences. Here, the decision-maker should be an expert in the domain. It is challenging to find the desired weights for different objectives. This section of this paper discusses the compromise programming approach that requires no pre-defined decision-maker preferences.

The problem of $4 \times G$ objective functions is complicated for decision-makers to define the weights corresponding to each lecturer. There are many proposed methods to solve multi-objective problems. Zeleny [25] introduced the ideal solution defined as the best-compromise solution that is the nearest to perfection. Ngo et al. [26–28] applied compromise programming to solve the problem of the binary objective in team selection, where they introduced the idea point E and try to find the solution that has minimum distance to E. Mahmudova used a variant of compromise programming called TOPSIS to identify the criteria and alternatives for software. They chose an alternate variant that has to be at the shortest Euclidean distance from the positive ideal solution and the farthest Euclidean distance from the negative ideal solution. The value of best and worst alternatives to software efficiency was found using the estimates of professional programmers [29]. Xu et al. determined the best compromise solution using a linear fuzzy membership function that represents the degree of achievement of an objective function as a value between 0 and 1. The best compromise solution is the one that archived the highest value of the normalized membership function μ_k calculated at the k -th solution [30]. Wei and Tian used a fuzzy statistic algorithm to select the best compromise solutions after obtaining Pareto-optimal solutions [31].

When the decision-maker stands in the view of lecturers, they declare their preferences on subjects and time-slots. It is hard to find the solution to archive the best, but we can define the best schedule they expect. The only goal left is to find a solution that is closest to this predefined point. The question we may ask decision-maker and predictable answer for them is as follows:

- How much faculty satisfaction on preferred time-slot and skill is good? Ideally, what they receive should be what they expect.

The decision-maker mostly provides this pair of the above question and answer for the time-slot, skill, number of courses, and part of the days they have to work. The objective function is now expressed as follow:

- Denote $E \in \mathbb{R}^{G \times (T+2)} = \{E_1, E_2, \dots, E_G\}$ is the matrix of idea timetable.

Where $E_g = \{E_{g,1}, E_{g,2}, \dots, E_{g,T}, E_{g,T+1}, E_{g,T+2}\}$ is the vector of expected timetable for lecturer g -th, such that

$$E_{g,j} = \begin{cases} \max_{h=1 \dots H | t_h = j} (a_{sh,g} * b_{j,g}) & \text{if } j \leq T \\ \text{norm}(D_g) & \text{if } j = T + 1 \\ \sqsupset & \text{otherwise} \end{cases}$$

The *norm* denotes the normalization function, \sqsupset is max rating for the number of parts of the days that a lecturer has to work.

- Let F is the matrix of the solution. $F \in \mathbb{R}^{G \times (T+2)} = \{F_1, F_2, \dots, F_G\}$ where $F_g = \{F_{g,1}, F_{g,2}, \dots, F_{g,T}, F_{g,T+1}, F_{g,T+2}\}$ is the vector of final timetable for lecturer g -th, such that:

$$F_{g,j} = \begin{cases} \sum_{h=1 | t_h=j}^H x_{h,g} * a_{sh,g} * b_{j,g} & \text{if } j \leq T \\ \text{norm}\left(\sum_{h=1}^H x_{h,g}\right) & \text{if } j = T + 1 \\ \text{pod}\left(\{x_{h,g} | h = 1 \dots H\}\right) & \text{otherwise} \end{cases}$$

- Q is the matrix of the worse possible solution. $Q \in \mathbb{R}^{G \times (T+1)} = \{Q_1, Q_2, \dots, Q_G\}$ where $Q_g = \{Q_{g,1}, Q_{g,2}, \dots, Q_{g,T}, Q_{g,T+1}, Q_{g,T+2}\}$ is the vector of the worse timetable for lecturer g -th, such that:

$$Q_{g,j} = \begin{cases} \begin{cases} 0 & \text{if } ss = 1 \dots SS | tm_{ss} = j \\ \max_{ss=1 \dots SS | tm_{ss}=j} (a_{sb_{ss},g} * b_{j,g}) * 2 > \text{max_rating} & \text{if } j \leq T \\ \text{max_rating} & \text{Otherwise} \end{cases} \\ \begin{cases} 0 & \text{if } D_g * 2 > T \\ T^2 & \text{Otherwise} \\ 0 & \text{otherwise} \end{cases} & \text{if } j = T + 1 \end{cases}$$

The original multi-objective functions (O1), (O2), (O3), and (O4) are rewritten in the form of compromise problem (CP):

$$\begin{aligned} \text{maximize(obj)} &= \left(1 - \frac{\text{distance}\left(\begin{bmatrix} E_1, E_2, \dots, E_G \end{bmatrix}, \begin{bmatrix} F_1, F_2, \dots, F_G \end{bmatrix}\right)}{\text{distance}\left(\begin{bmatrix} E_1, E_2, \dots, E_G \end{bmatrix}, \begin{bmatrix} Q_1, Q_2, \dots, Q_G \end{bmatrix}\right)}\right) \\ &= 1 - \sqrt{\frac{\sum_{i=1}^G \sum_{j=1}^{T+1} (E_{i,j} - F_{i,j})^2}{\sum_{i=1}^G \sum_{j=1}^{T+1} (E_{i,j} - Q_{i,j})^2}} \end{aligned}$$

3. Proposed Algorithm

3.1. Introduction to Genetic Algorithm

The Genetic Algorithm [32,33] is a population-based metaheuristic method extensively used in scheduling problems. It searches a solution space for the optimal solution to a problem. This search is done in a fashion that mimics the operation of evolution. In essence, a “population” of possible solutions formed, and new solutions are created by “breeding” the best individual from the population’s members to build a new generation. When the algorithm converged after several generations, the best solution returned. Genetic algorithms are particularly useful for problems where it is extremely difficult or impossible to get an exact answer or severe problems where a correct solution may not be required. They offer an exciting alternative to the typical algorithmic solution methods and are highly customizable. This notion can apply to a search problem. We consider a set of solutions for a challenge and select the set of best ones out of them. There are five phases considered in

a genetic algorithm. This study introduces a version of the Genetic Algorithm to solve the MOP model Compromise Programming approach with a new added phase called “repair” to correct the errors. The flow of the proposed scheme is displayed in Figure 2.

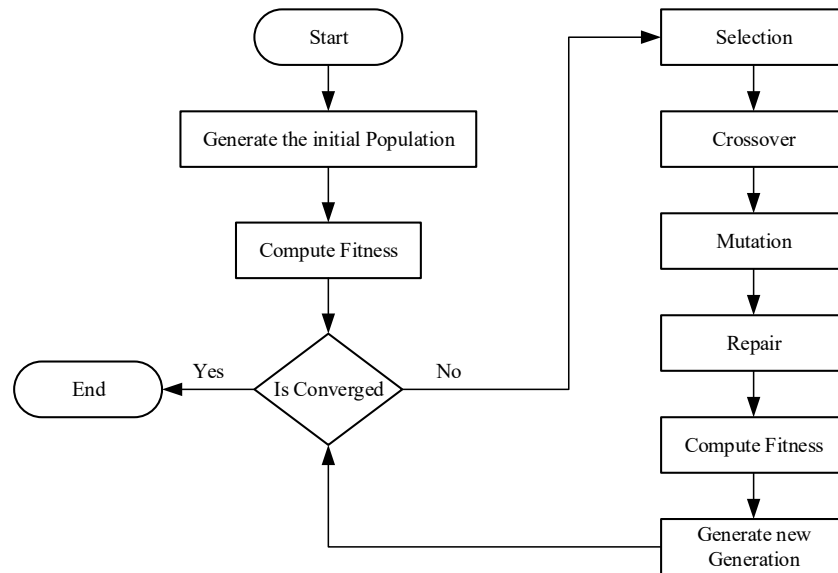


Figure 2. Basic workflow of the proposed Genetic Algorithm’s scheme.

3.2. Genetic Algorithm Scheme

3.2.1. Genetic Representation

Chromosome is represented as a matrix of G rows and T columns, rows i -th represent the section assignment for lecturer i -th. Cell (g, t) contains section lecturer g -th assigned to time-slot t -th, or 0 if lecturer g -th is not assigned to any section at the time-slot t -th.

3.2.2. Fitness Function

The fitness function contains two components: the penalty function and the objective function. While the objective function focuses on optimizing the lecturer’s satisfaction, the penalty function deals with constraints. We separate constraints into two groups, group 1st includes constraint (H4) handled by penalty function and group 2nd includes the remaining constraints (H1), (H2), (H3) handled by repair mechanism described in Section 3.2.4. So, we have the fitness function:

$$f = w_{pen} * pen + w_{obj} * obj$$

where pen , obj , w_{pen} , w_{obj} denote penalty function, objective function, penalty function weight, and objective function weight respectively. We normalize the penalty function, objective function, and weights to 0–1 range. So we have the constraints: $0 \leq w_{pen}, w_{obj} \leq 1$, and $w_{pen} + w_{obj} = 1$. Let V be the number of lecturer violate constraint (H4), the penalty function is normalized as follows:

$$pen = \frac{1}{1 + V}$$

3.2.3. Algorithm Operations

Denote:

- U represents the size of the population.
- $P^e = \{p_i^e \mid i = 1 \dots U\}$ as the population at generation e -th.
- p_i^e as the individual i -th of the population at generation e -th, represented as chromosome matrix described in Section 3.2.1. $p_{i,n,m}^e$ denotes the value of the cell at row n -th and column m -th.

- ∂_t as the set of sections that learn at time-slot t -th.
- φ as the tournament size for selection.
- B as the mutation rate

Step 1: Generate the initial population: Columns k -th of an individual p_i^e contain ∂_k and exactly $G - |\partial_k|$ number 0. So, for each column $k \mid k = 1 \dots T$, fill ∂_k and $G - |\partial_k|$ number 0 to that column, and shuffle its element to ensure the randomness of the initialized population. After filling all columns to chromosome matrix, apply repair operator to ensure the created chromosome respects constraints (H1), (H2), and (H3).

Step 2: Selection: we implemented the selection process based on Tournament Selection [34]. Randomly select φ individuals from P^e and perform a tournament that return the best individuals based on fitness value among them.

Step 3: Crossover: p_{father}^e and p_{mother}^e are the parents to crossover. The set of numbers in each column of p_{father}^e and p_{mother}^e is permutation of each other, so we can choose any ordered crossover method to apply. Partially mapped crossover (PMX) [35] is one of the most effective crossover techniques for ordered list, so it is chosen in this study.

Step 4: Mutation: For each individual p_i^e , have rate B to swap only once for any two elements in any column. Similar to generating initial population process, the created chromosome after performing crossover and mutation must be applied to repair operator to ensure there are no invalid results during the processing.

3.2.4. Repair Process

Input a chromosome matrix p which may violate constraints (H1), (H2), and (H3), genetic repair operator rearrange elements in p so that new chromosome p' satisfies all these constraints. Moreover, p' should retain as many p 's genes as possible. The purpose we combine constraints (H1), (H2), (H3) into one group is because it is very easy to convert them into the maximum matching problem in bipartite graph. In this study, we use the Hopcroft–Karp algorithm [36], a polynomial algorithm to find the maximum matching. The repair process is performed in three steps as follows:

Step 1: Build a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with vertex set $\mathcal{V} = \mathcal{X} \cup \mathcal{Y}$, \mathcal{X} represents vertex set of $G \times T$ items for each lecturer in each timeslot, \mathcal{Y} represents vertex set of H items represent for sections. For each vertex $\mathcal{G} \in \mathcal{X}$ (\mathcal{G} is the vertex represents for lecturer g -th at timeslot t -th), $h \in \mathcal{Y}$ (h is the vertex represents for section h^{th}), we add an edge from \mathcal{G} to h if and only if $t = t_h$, $a_{s_h, g} > 0$ and $b_{t, g} > 0$.

Step 2: For each lecturer $g \mid g = 1 \dots G$ at each timeslot $t \mid t = 1 \dots T$, pair the vertex $u \in \mathcal{X}$ (u is the vertex represents for lecturer g -th at timeslot t -th) to vertex $v \in \mathcal{Y}$ (v is the vertex represents for section $p_{g,t}^{th}$) if $p_{g,t} > 0$ and the pairing does not violate any constraints in (H1), (H2), (H3). This step aims to retain the good genes from p .

Step 3: Apply the Hopcroft–Karp algorithm to graph \mathcal{G} built in step 1 with pre matching in step 2, we get the final matching which represents the repaired chromosome p' .

4. Experiment and Result

To evaluate the proposed model and algorithm, we use the data collected in the spring semester of 2020 of the Computing Fundamental department at FPT University. A total of $H = 139$ sections of $S = 17$ subjects were assigned to $G = 27$ lecturers. The pod function's returned values were collected depending on the different business settings. Table 1 illustrates the definition of the timeslots at the institution. The merged cells described that two real-life slots are marked as the same time slot in the mathematical model.

We construct a fuzzy function pod to fit the definition of the timeslots as the following algorithms. Our goal is to determine the number of sessions per day that the instructor must be in college and rate it. A good schedule that is matching the number of sections to teach with the number of part of days that lecturer has to present at the school gain 100% satisfaction (\square points). Satisfaction is inversely proportional to the instructor's waiting time. For some situations, when hiring qualified lecturers is not easy work, a compact schedule in the aspect

of time may give more flexibility to the human resources organizing process.

Function: <i>pod</i>	
Input: $\{x_{h,g} \mid h = 1 \dots H\}$	
1:	$\lceil = 100$
2:	$r = \text{NumPod}(\{x_{h,g} \mid h = 1 \dots H\})$
3:	$n = \sum_{h=1}^H x_{h,g}$
4:	If $(1 \leq n \leq 3)$ and $(r = 1)$ Return \lceil
5:	If $(1 \leq n \leq 3)$ and $(r = 2)$ Return $\lceil/5$
6:	If $(1 \leq n \leq 3)$ and $(r \geq 3)$ Return 0
7:	If $(4 \leq n \leq 6)$ and $(r = 2)$ Return \lceil
8:	If $(4 \leq n \leq 6)$ and $(r = 3)$ Return $\lceil/5$
9:	If $(4 \leq n \leq 6)$ and $(r = 4)$ Return 0
10:	If $(1 \leq n \leq 3)$ and $(r = 1)$ Return \lceil
11:	If $(7 \leq n \leq 8)$ and $(r = 3)$ Return \lceil
12:	If $(7 \leq n \leq 8)$ and $(r = 4)$ Return $\lceil/2$
13:	If $(9 \leq n \leq 10)$ Return \lceil


The <i>NumPod</i> function defined as:	
Function: <i>NumPod</i>	
Input: $\{x_{h,g} \mid h = 1 \dots H\}$	
1:	$\text{Num} = 0$
2:	If $(\sum_{t_h \in \{M1, M2, M3\}} x_{h,g} \geq 1)$ Then $\text{Num} = \text{Num} + 1$
3:	If $(\sum_{t_h \in \{E1, E2, E3\}} x_{h,g} \geq 1)$ Then $\text{Num} = \text{Num} + 1$
4:	If $(\sum_{t_h \in \{M4, M5\}} x_{h,g} \geq 1)$ Then $\text{Num} = \text{Num} + 1$
5:	If $(\sum_{t_h \in \{E4, E5\}} x_{h,g} \geq 1)$ Then $\text{Num} = \text{Num} + 1$
6:	Return Num

We built a webpage to collect lecturer preferences of the subjects, time-slots, and the number of time-slots they want to teach, as shown in Figure 3. The preferences gathering process needs to be done before starting the scheduler triggered. The lecturers use their personal accounts to make declarations with the system. In this experiment, we set the preferences received the values in the range of $a_{s,g} \in [0 \dots 5]$ and $b_{t,g} \in [0 \dots 5]$.

The developed system described in this report was deployed on a computer configured as follows: Processor: Intel(R) Xeon(R) CPU X5650 @2.67 GHz (4 CPUs), ~2.3 GHz; Memory: 8096 MB RAM; all code implemented in java 8.

The designed genetic algorithm operated based on several parameters. They have a significant influence on the results of the algorithm. In this section, we describe how the values of this parameter are selected. To select the most suitable parameters for the genetic algorithm, we execute the algorithm multiple times. Observed effects on the corresponding MOP approaches are listed in Table 2.

The tested ranges of the importance of the parameters show good results. The small tournament size makes the crossover loses diversity. It negatively affects the algorithm results as well as the time of convergence. The tournament size = 7 seems to generate good results for the scalarizing approach, and tournament size > 9 increases the processing time even though it maintains an excellent fitness value. Population size > 100 gets worse for both fitness values and processing time. Based on the observed results, we selected the parameter set to run the algorithm according to Table 3.



Timetable

Manage

Expected

DSST

Arrange

Request

Expected

D

SELECT TERM :

Summer 2020

Edit

YOU HAVE FILLED OUT EXPECTED FOR THIS TERM !

SUBJECT / PREFERENCE LEVEL :

	0	1	2	3	4	5
CSD201	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
CSI101	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
CSI103	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
DBI202	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
DBW301	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
FUN131B	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
JFE301	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
NWC202	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
OSG202	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
PRE201	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
PRF192	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
PRJ311	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
PRJ321	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
PRN292	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
PRO192	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
WED201C	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

SLOT / PREFERENCE LEVEL :

	0	1	2	3	4	5
M1	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
M2	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
M3	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
E1	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
E2	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
E3	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
M4	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
M5	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
E4	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
E5	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Note

M1: slot 1 Mon,Tue,Fri
M2: slot 2 Mon,Tue,Fri
M3: slot 3 Mon,Tue,Fri
E1: slot 4 Mon,Tue,Fri
E2: slot 5 Mon,Tue,Fri
E3: slot 6 Mon,Tue,Fri
M4: slot 1,2 Tue | 1 Thursday
M5: slot 3 Tue | 2,3 Thursday
E4: slot 4,5 Tue | 1 Thursday
E5: slot 6 Tue | 5,6 Thursday

0,1,2,3,4,5:

- Level of Lecturer's preference for slot/ subject.
- The higher number, the higher preference.
- Select 0 if Lecturer cannot teach this subject / slot.

NUMBER OF CLASS (1-10)

10

NOTE

abc

Figure 3. Webpage to collect the preferences of a particular lecturer on the subjects and time-slots.

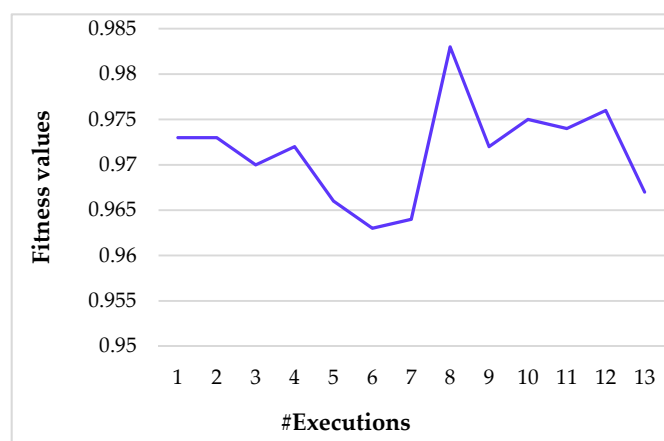
Table 2. The observation result of the algorithm for each set of parameters.

Param	Value	Observation Results
mutation	0.9–1	Stable results, processing time increased slightly
	0.5–0.8	Stable results, processing time increased
	0–0.5	Stable results, stable time execution
tournament size	2	Results decreased slightly, stable time execution
	3	Stable results, processing time increased
	5	Stable results, stable time execution
	7	Stable results, processing time decreased
	9	Stable results, processing time decreased
population size	100–150	Stable results, processing time increased
	151–200	Stable results decreased, processing time increased

Table 3. Parameters to run the algorithm.

Parameter	B	U	φ	Stop after	w_{obj}	w_{pen}
Value	0.4	100	7	30	0.3	0.7

To evaluate the proposed algorithm. We have run the algorithm multiple times with the same initial value. Figure 4 shows the fitness values over 13 executions. It shows that the result is nearing expected values (approximately 1) on tested data. The average time execution is around 33 s, as shown in Figure 5. In fact, the teaching assignment process accounts for an average of 1 working day of the head of department. In many cases when the number of sections is too large, it is even more time-consuming.

**Figure 4.** Fitness values of GA over several executions.**Figure 5.** Execution time of GA over several executions.

The fitness values change during each generation of the Genetic Algorithm is shown in Figure 6. After about the first 20 generations, the fitness value came very close to the convergence value. The proposed model allows faculty preferences for four aspects: skills, timeslots, number of classes, and working time. It considers more aspects of stakeholders' needs than the simple "sum of favorites on the particular wish of lecturers" model introduced by previous research [17,21]. We use a non-preference approach for the multi-objective problem. Compromise programming gives a satisfactory answer in cases where there is not any priority assigned. The lecturer's satisfaction degrees corresponding to the groups of objective functions were obtained by executing GA displayed in Figure 7. It observed that teachers who are registered to teach many subjects could guide in many different time frames and naturally prioritize various topics. Meanwhile, with the target

function's current scoring: 100%~5 stars of subjects * 5 stars of time slots, which leads to those who can teach few items or more constrained about time constraints may receive a less-satisfied schedule.

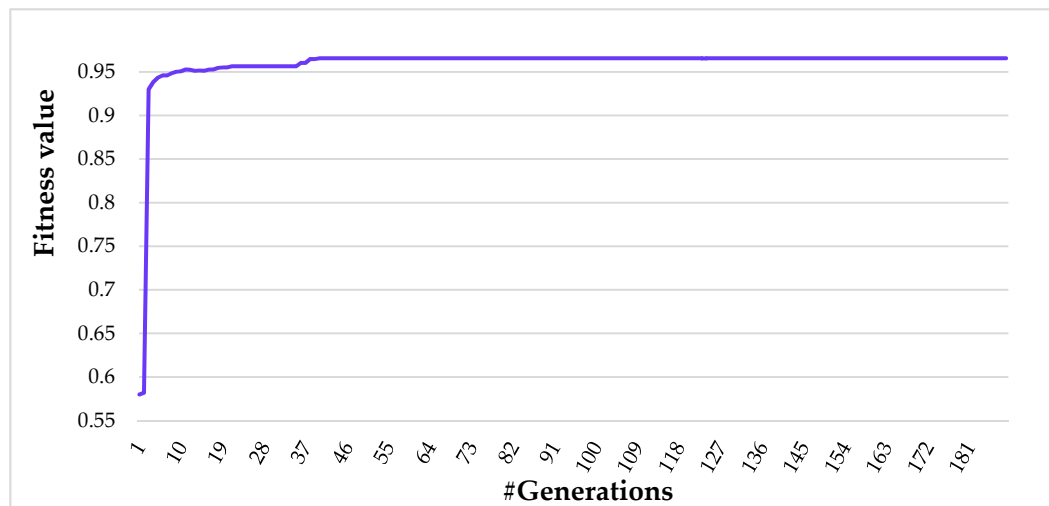


Figure 6. Fitness values changing over generations.

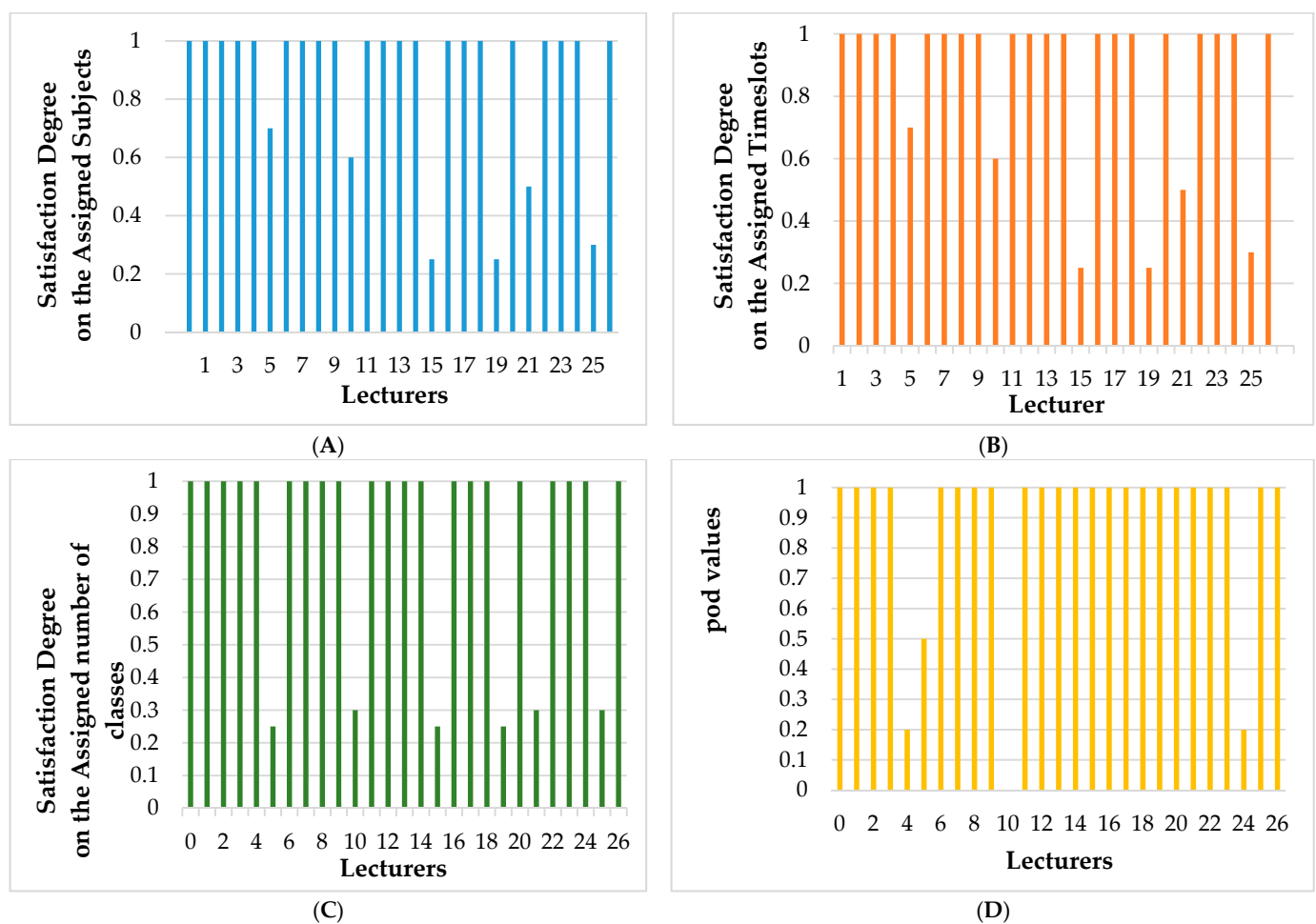


Figure 7. Satisfaction degrees of the lecturers. (A) Satisfaction degree on the assigned subjects. (B) Satisfaction degree on the assigned timeslots. (C) Satisfaction degree on the assigned number of classes. (D) The returned values of the part of the day function (pod).

We illustrate the optimal solution in Figure 8. A directed graph is used to represent matching between instructor and sections. Each section node will only associate with one node trainer. In this situation, the arrow comes from a trainer node and points to the corresponding section node.

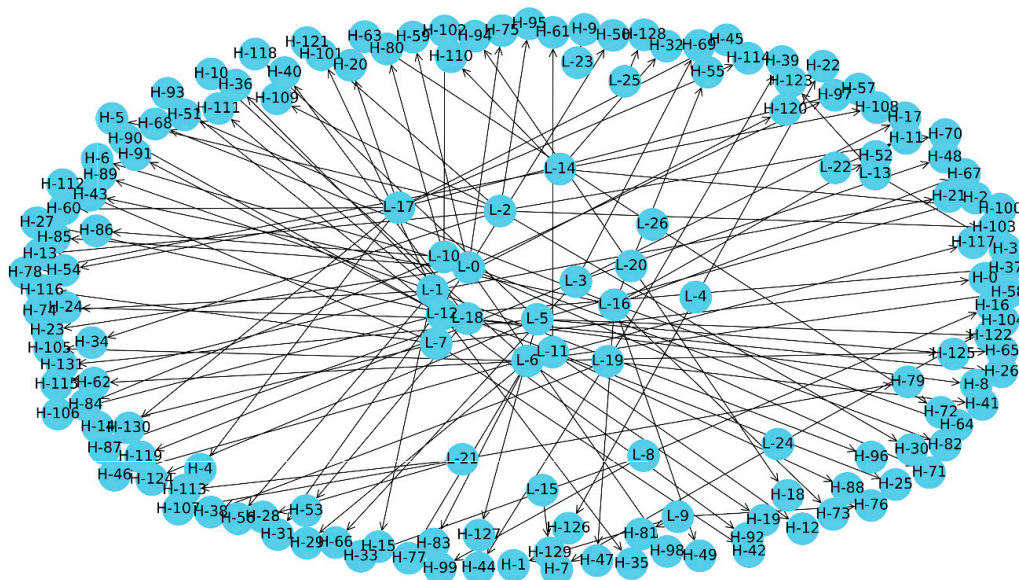


Figure 8. Obtained solution visualized by a directed graph.

The Genetic Algorithm (and also other approximation algorithms) does not guarantee to find the global solution. The obtained solutions may be local optima. Since our model is different from previous models, it is not feasible to compare the proposed algorithm with the available settings. However, we can evaluate whether the algorithm works well in terms of a smaller search space. We extracted a data set from 5 lecturers, and 12 sections then found the global solution using exhaustive search. The proposed scheme's comparison results with the implementation of the Brute Force algorithm are shown in Table 4, and the optimal solution is illustrated in Figure 9.

Table 4. Comparison between Genetic Algorithm and Brute Force Algorithm.

Algorithm	Fitness Values	Time Execution (s)
Brute Force	0.95919	468
Genetic Algorithm	0.95919	0.55

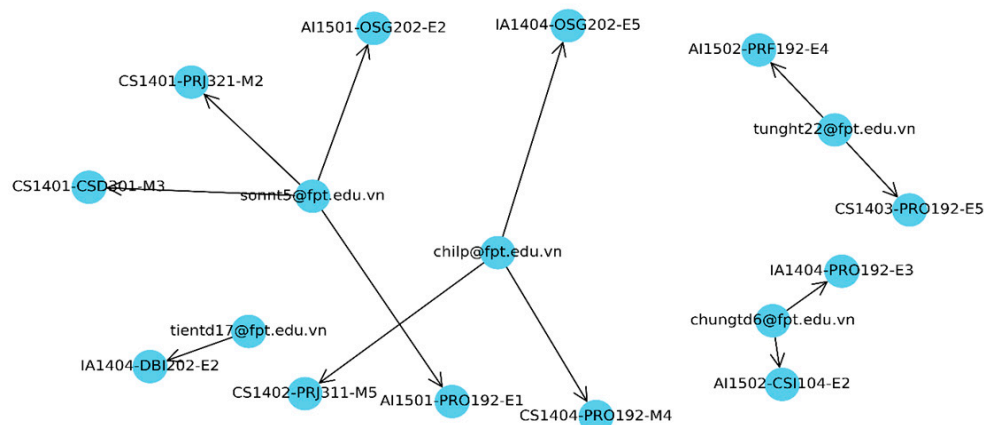


Figure 9. Global Solution obtained by Brute Force.

Decision-maker may have their customization on the provided schedule in this situation. To support them, modify the plan quickly, we design a web page to help drag and drop, as shown in Figure 10. The decision-maker can choose any course and assign it to another instructor by dropping the item in the corresponding line. The information systems part plays a vital role in compensating for the shortcomings of the proposed algorithm.

The screenshot displays the 'Arrange' web interface for customizing a timetable. The interface includes a sidebar with navigation options (Timetable, Manage, Expected, DSST, Arrange, Request), a top header with the FPT University logo, and a main content area. The main area has filters for Summer 2020, Select Teacher, Select Class, Select Room, and Select Subject, with a Search button. Below these are 'Timetable Modify' controls for Subject (DBI202), Room (AL-L202), and Lecturer (sonnt5), with buttons for Edit, Swap Timetable, Lecturer Swap, Request Confirm, and Request Lecturer Confirm. The central part is a grid table showing the schedule for rooms AL-L202 through AL-L207 across various subjects and lecturers.

	M1	M2	M3	E1	E2	E3	M4	M5	E4	E5
AL-L202	SE1424 DBI202 sonnt5	SE1424 PRO192 hapt27		SE1425 DBI202 sonnt5	SE1425 PRO192 hapt27					
AL-L203		SE1426 DBI202 ngaDTT22	SE1426 PRO192 chlp		SE1427 DBI202 huongnt7	SE1427 PRO192 binhmv2				
AL-L204			SE1428 DBI202 ThieuVV2			SE1429 DBI202 ngaDTT22	SE1428 PRO192 thopn3		SE1429 PRO192 sonnt5	
AL-L205							SE1430 PRO192 thuyntb3	SE1430 PRO192 caupd	SE1431 DBI202 huongnt7	SE1431 PRO192 sonnt5
AL-L206	SE1432 PRO192 hapt27			SE1433 PRO192 ThieuVV2				SE1432 DBI202 sonnt5		SE1433 DBI202 huongnt7
AL-L207	SE1434 DBI202 ngaDTT22	SE1434 PRO192 ThieuVV2		SE1435 DBI202 ngaDTT22	SE1435 PRO192 sonnt5					

Figure 10. The webpage allows the decision-maker to customize the generated schedule.

5. Conclusions

In this study, we have proposed a multi-objective optimization model for the assignment task. The proposed model satisfies the lecturers' preferences regarding skills, time, and the number of jobs while ensuring related constraints. Our model was applied to the FPT University lecturers scheduling problem and defined a generic solution for multi-objective task assignment problems. We use compromise programming to turn the multi-objective problem into a single-objective problem. Although in the preferred approach, users can set different values for each weight of the target function. It is flexible, but in a multi-dimensional space, the visualization of the results corresponding to a parameter set is difficult. It leads decision-makers to explore parameter sets in an ample search space. On the other hand, a compromise model is a single-shot solution for decision-makers. It avoids them having to define preference information for the objectives. The model itself has found a way to the best. However, the low use of parameters reduces the ability to interact with the model of a decision-maker. The proposed scheme for genetic algorithm shows that it works effectively; the repair step has removed all binding violation solutions without affecting crossovers' diversity. Shortly, we are looking to build an integrated model with lecturers and students' scheduling simultaneously. Refining the parameters of the algorithm is also a job in our plan.

Author Contributions: Conceptualization, S.T.N., B.N.A. and J.J.; formal analysis, B.N.A.; funding acquisition, B.N.A.; investigation, S.T.N.; methodology, S.T.N.; resources, S.T.N. and B.N.A.; software, S.T.N. and B.N.A.; supervision, J.J. and I.A.A.; validation, J.J. and I.A.A.; writing—original draft, S.T.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by FPT University, grant number DHFPT/2020/12.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Andrade, P.R.L.; Steiner, M.T.A.; Góes, A.R.T. Optimization in timetabling in schools using a mathematical model, local search and Iterated Local Search procedures. *Gestão Produção* **2019**, *26*, e3421. [\[CrossRef\]](#)
2. Lemos, A.; Melo, F.S.; Monteiro, P.T.; Lynce, I. Room usage optimization in timetabling: A case study at Universidade de Lisboa. *Oper. Res. Perspect.* **2018**, 100092. [\[CrossRef\]](#)
3. Ghiani, G.; Manni, E.; Romano, A. Training offer selection and course timetabling for remedial education. *Comput. Ind. Eng.* **2017**, *111*, 282–288. [\[CrossRef\]](#)
4. Vermuyten, H.; Lemmens, S.; Marques, I.; Beliën, J. Developing compact course timetables with optimized student flows. *Eur. J. Oper. Res.* **2016**, *251*, 651–661. [\[CrossRef\]](#)
5. Babaei, H.; Karimpour, J.; Hadidi, A. A survey of approaches for university course timetabling problem. *Comput. Ind. Eng.* **2015**, *86*, 43–59. [\[CrossRef\]](#)
6. Pentico, D.W. Assignment problems: A golden anniversary survey. *Eur. J. Oper. Res.* **2007**, *176*, 774–793. [\[CrossRef\]](#)
7. Daskalaki, S.; Birbas, T.; Housos, E. An integer programming formulation for a case study in university timetabling. *Eur. J. Oper. Res.* **2004**, *153*, 117–135. [\[CrossRef\]](#)
8. Feng, X.; Lee, Y.; Moon, I. An integer program and a hybrid genetic algorithm for the university timetabling problem. *Optim. Methods Softw.* **2016**, *32*, 625–649. [\[CrossRef\]](#)
9. Nouri, H.E.; Driss, O.B. MATP: A Multi-agent Model for the University Timetabling Problem. In *Software Engineering Perspectives and Application in Intelligent Systems*; Silhavy, R., Senkerik, R., Oplatkova, Z., Silhavy, P., Prokopova, Z., Eds.; Advances in Intelligent Systems and Computing; Springer: Cham, Switzerland, 2016; Volume 465. [\[CrossRef\]](#)
10. Nouri, H.E.; Driss, O.B. Distributed model for university course timetabling problem. In Proceedings of the 2013 International Conference on Computer Applications Technology (ICCAT), Sousse, Tunisia, 20–22 January 2013. [\[CrossRef\]](#)
11. Malik, B.B.; Nordin, S.Z. Mathematical model for timetabling problem in maximizing the preference level. In Proceeding of the 25th National Symposium on Mathematical Sciences (Sksm25): Mathematical Sciences as the Core of Intellectual Excellence, Pahang, Malaysia, 27–29 August 2017; AIP Conference Proceedings. p. 020037. [\[CrossRef\]](#)
12. Santos, H.G.; Uchoa, E.; Ochi, L.S.; Maculan, N. Strong bounds with cut and column generation for class-teacher timetabling. *Ann. Oper. Res.* **2012**, *194*, 399–412. [\[CrossRef\]](#)
13. Dorneles, Á.P.; de Araújo, O.C.B.; Buriol, L.S. A fix-and-optimize heuristic for the high school timetabling problem. *Comput. Oper. Res.* **2014**, *52*, 29–38. [\[CrossRef\]](#)
14. Hmer, A.; Mouhoub, M. Teaching Assignment Problem Solver. *Lect. Notes Comput. Sci.* **2010**, 298–307. [\[CrossRef\]](#)
15. Kuhn, H.W. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.* **1955**, *2*, 83–97. [\[CrossRef\]](#)
16. Fisher, M.L.; Jaikumar, R.; Wassenhove, L.N.V. A multiplier adjustment method for the generalized assignment problem. *Manag. Sci.* **1986**, *32*, 1095–1103. [\[CrossRef\]](#)
17. Lewis, R. A survey of metaheuristic-based techniques for University Timetabling problems. *OR Spectr.* **2007**, *30*, 167–190. [\[CrossRef\]](#)
18. Muthuraman, S.; Venkatesan, V.P. A Comprehensive Study on Hybrid Meta-Heuristic Approaches Used for Solving Combinatorial Optimization Problems. In Proceedings of the 2017 World Congress on Computing and Communication Technologies (WCCCT), Tiruchirappalli, India, 2–4 February 2017.
19. Sigl, B.; Golub, M.; Mornar, V. Solving timetable scheduling problem using genetic algorithms. In Proceedings of the 25th International Conference on Information Technology Interfaces (ITI 2003), Cavtat, Croatia, 19 June 2003; pp. 519–524.
20. Savić, A.; Tošić, D.; Marić, M.; Kratica, J. Genetic algorithm approach for solving the task assignment problem. *Serdica J. Comput.* **2008**, *2*, 267–276.
21. Sapru, V.; Reddy, K.; Sivaselvan, B. Time table scheduling using Genetic Algorithms employing guided mutation. In Proceedings of the 2010 IEEE International Conference on Computational Intelligence and Computing Research, Coimbatore, India, 28–29 December 2010; pp. 1–4.
22. Yang, S.; Jat, S.N. Genetic algorithms with guided and local search strategies for university course timetabling. *Syst. Man Cybern. Part C Appl. Rev.* **2011**, *41*, 93–106. [\[CrossRef\]](#)
23. Corne, D.; Ross, P.; Fang, H. Evolving timetables. In *The Practical Handbook of Genetic Algorithms*; Chambers, L.C., Ed.; CRC: Boca Raton, FL, USA, 1995; Volume 1, pp. 219–276.
24. Hwang, C.-L.; Masud, A.S.M. *Multiple Objective Decision Making, Methods and Applications: A State-of-the-Art Survey*; Springer: Berlin/Heidelberg, Germany, 1979; ISBN 978-0-387-09111-2.
25. Zeleny, M. Compromise Programming. In *Multiple Criteria Decision Making*; Cochrane, J.L., Zeleny, M., Eds.; University of South Carolina Press: Columbia, SC, USA, 1973; pp. 262–301.
26. Son, N.T.; Thanh, L.V.; Duong, T.B.; Anh, B.N. A decision support tool for cross-functional team selection: Case study in ACM-ICPC team selection. In *Proceedings of the 2018 International Conference on Information Management & Management Science (IMMS'18)*; ACM: New York, NY, USA, 2018; pp. 133–138. [\[CrossRef\]](#)

-
27. Son, N.T.; Thuy, T.T.; Anh, B.N.; van Dinh, T. DCA-Based Algorithm for Cross-Functional Team Selection. In *Proceedings of the 2019 8th International Conference on Software and Computer Applications (ICSCA'19)*; ACM: New York, NY, USA, 2019; pp. 125–129. [[CrossRef](#)]
 28. 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. [[CrossRef](#)]
 29. Mahmudova, S. Application of the TOPSIS method to improve software efficiency and to optimize its management. *Soft Comput.* **2020**, *24*, 697–708. [[CrossRef](#)]
 30. Xu, X.; Hu, Z.; Su, Q.; Xiong, Z. Multiobjective Collective Decision Optimization Algorithm for Economic Emission Dispatch Problem. *Complexity* **2018**, *2018*, 1027193. [[CrossRef](#)]
 31. Wei, W.; Tian, Z.-Y. An improved multi-objective optimization method based on adaptive mutation particle swarm optimization and fuzzy statistics algorithm. *J. Stat. Comput. Simul.* **2017**, 1–14. [[CrossRef](#)]
 32. Van Veldhuizen, D.A.; Lamont, G.B. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evol. Comput.* **2000**, *8*, 125–147. [[CrossRef](#)] [[PubMed](#)]
 33. Thede, S.M. An Introduction to Genetic Algorithms. *J. Comput. Sci. Coll.* **2004**, *20*.
 34. Miller, B.; Goldberg, D. Genetic Algorithms, Tournament Selection, and the Effects of Noise. *Complex Syst.* **1995**, *9*, 193–212.
 35. Otman, A.; Jaafar, A. A comparative study of adaptive crossover operators for genetic algorithms to resolve the travelling salesman problem. *Int. J. Comput. Appl.* **2011**, *31*, 49–57.
 36. Hopcroft, J.E.; Karp, R.M. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.* **1973**, *2*, 225–231. [[CrossRef](#)]