Article

A Decision Support Tool for Multi-Objectives Teaching Assignment Problem

Ngo Tung Son[1,*], Bui Ngoc Anh[1], Jafreezal Jaafar[2]

[1] Information and Communication Technology Department, FPT University, Hanoi 100000, Vietnam; sonnt69@fe.edu.vn; anhbn5@fe.edu.vn;

[2] Department of Computer and Information Sciences, Center for Research in Data Science, Universiti Teknologi PETRONAS, Tronoh 32610, Malaysia; jafreez@utp.edu.my;

* 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 assigning task to the lecturer is an example of the timetabling problem, a class of scheduling. This study introduces a mathematical model to assign fixed tasks (the time and required skills to be fixed) 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 optimal makes decisions require the trade-off presence of trade-offs between two or more conflicting objectives. To solve this, we use the Compromise Programming approach to multi-objective programming. We then proposed the new version of the Genetic Algorithm to solve the introduced model. Finally, the model and algorithm tested with real scheduling data collected at the Computing Fundamental Department, FPT University, Hanoi, Vietnam.

Keywords: Timetabling, Task Assignment, MOP, Combinatory Optimization, Compromise Programming, Genetic Algorithm.

1. Introduction

1.1. Background

The need to optimize types of resources is as much a requirement in training organizations as in any other kind of institution. 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 meant to be a group 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 attached. The university usually performs a scheduling task before a semester begins [1, 2, 3, 4, 5]. The scheduling / timetabling problem comes in many forms. Each of them requires a different strategic approach.

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 our 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 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 problem becomes an instance of the teaching assignment problem [6].

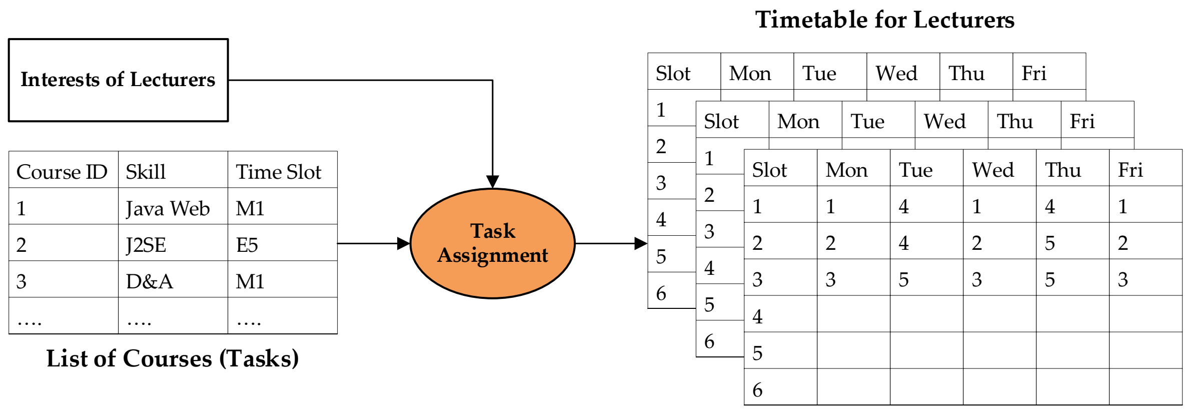

Figure 1: The teacher assignment problem.

The input data for the decision-making stage described as Every course/class established for a particular subject and took place in 30-time slots (equivalent to 45 hours of study, 1-time slots equals 1.5 hours). A student can join a maximum of ten classes, but only one class per time-slot. The duration of a semester is ten weeks. Every class has three slots of the same subject per week and must not occur for two consecutive days, giving the student time to prepare for the next sections. Each semester the university opens about 1000 classes. Lecturers can teach a maximum of 6 slots per day. Table 1 illustrates ten available time-slots for a teacher/student for every week in the semester. The system assigns these courses to the lecturers based on their skills and expectations (more detail in section 2).

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

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

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 they try 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].

The task assignment problems exist in many different forms. While some of them like the classical problem have polynomial-time solutions [9], others are NP-hard combination optimization problems [10] that requires approximation approaches, especially metaheuristic [11]. In [12], 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 [13]. They reviewed several algorithms, such as ant colony optimization, evolutionary computation, particle swarm optimization, etc. Related to the scheduling problem, many researchers used genetic algorithms, an evolutionary algorithm to solve scheduling problems, and task assignment problems [14,15,16]. 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 [17]. Yang develops an efficient hybrid genetic algorithm based on algorithms for the converted problem [8]. 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 work we perform is a stage in the automatic scheduling solution at FPT University and a new approach for the fixed-tasks assignment problem. The related researches to the scheduling may benefit from our study. Due to the fixed schedule that respects the business requirement, the considered events such as classes, time-slots, and subjects determined. Our mission is to arrange the available works for available human resources. We built a multi-objectives model that accesses each individual's level of interest assigned to the job, which still provides a binding compliance solution. Our proposed model covers more business requirements rather than previous works—the optimization model of the problem 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-maker groups, but Compromise Programming works extremely better if no preference is indicated. We have implemented a Genetic Algorithm that solves both optimal models of a target mentioned above—the detail of the implementation 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. Problem Formulation

2.1. Multi-Objective Task Assignment Problem

Many researchers have used the integer programming (IP) model to solve this problem, such as [1, 3]. 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} prefer 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 $s = 1 \dots S, g = 1..G$ 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 $t = 1 \dots T, g = 1..G$ 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.

In [18], Corne suggested some of timetabling constraints such as (1) Unary constraints, (2) Binary constraints, (3) Capacity constraints, (4) Event spread constraints, (5) Agent constraints. Based on those suggestions, we also define several constraints to the problem as follows:

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

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

- 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 the past, 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 [19, 20] 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. build a model for mapping the task to the lecturer that maximizes their preference on the time-slot [21]. 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 [22,23]. 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.

$$\max\left\{\sum_{h=1}^{H} x_{h,g} * a_{s_h,g}\right\} \quad \forall g = 1 \dots G \quad (01)$$

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

$$\max\left\{\sum_{h=1}^{H} x_{h,g} * b_{t_h,g}\right\} \quad \forall g = 1 \dots G \quad (02)$$

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

$$\min\left\{\left|\sum_{h=1}^{H} x_{h,g} - D_g\right|\right\} \quad \forall g = 1 \dots G \quad (03)$$

- Minimize the number of parts of the day, which lecturers have to 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):

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

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 [25]. 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*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 [26] introduced the ideal solution defined as the best-compromise solution that is the nearest to perfection. Ngo et al. [27, 28, 29] 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. 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 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
$$E_{g,j} = \begin{cases} \max\limits_{h=1..H|t_h=j} \left(a_{s_h,g} * b_{j,g}\right) & if\ j \le T \\ norm(D_g\ ) & if\ j = T+1 \\ ב & otherwise \end{cases}$$
  Where *norm* denotes the normalization function, ב 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:
$$F_{g,j} = \begin{cases} \sum\limits_{h=1|t_h=j}^{H} x_{h,g} * a_{s_h,g} * b_{j,g} & if\ j \le T \\ norm\left(\sum\limits_{h=1}^{H} x_{h,g}\right) & if\ j = T+1 \\ pod\left(\{x_{h,g} |\ h = 1..H\}\right) & otherwise \end{cases}$$

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

$$\min\left(distance([E_1, E_2, \dots, E_G], [F_1, F_2, \dots, F_G])\right) = \sqrt{\sum_{i=1}^{G}\sum_{j=1}^{T+2}\left(E_{i,j} - F_{i,j}\right)^2}$$

3. Proposed Algorithm

3.1. Introduction to Genetic Algorithm

The Genetic Algorithm [30] 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 displays in Figure 2.

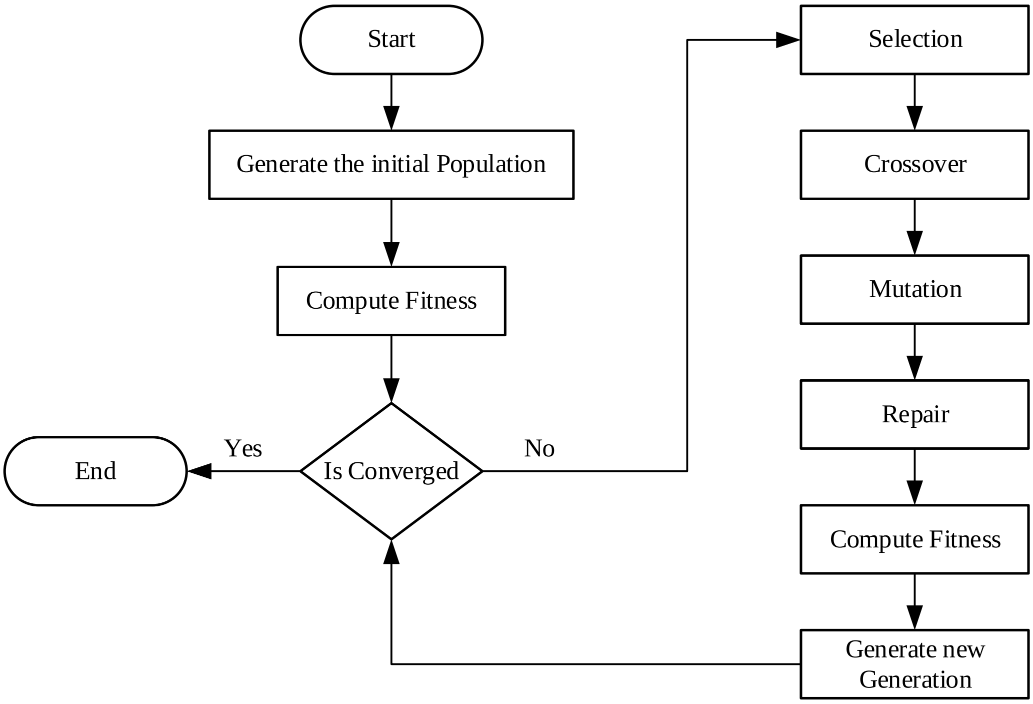

Figure 2: basic workflow of the proposed Genetic Algorithm's Scheme

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 at time-slot t^{th}, or 0 if lecturer g^{th} is not assigned to any section at time-slot t^{th}.

3.2.2. Fitness function

The fitness function contains two components: the penalty function and objective function. While the objective function focuses on optimizing lecturer's satisfaction, the penalty function deals with constraints. We separate constraints into 2 groups, group 1^{st} includes constraint (H4) handled by penalty function and group 2^{nd} 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 follow:

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

The objective function defined for Compromise Programming as:

$$1 - \frac{distance([E_1, E_2, \ldots, E_G], [F_1, F_2, \ldots, F_G])}{distance([E_1, E_2, \ldots, E_G], [Q_1, Q_2, \ldots, Q_G])} = 1 - \sqrt{\frac{\sum_{i=1}^{G}\sum_{j=1}^{T+1}\left(E_{i,j} - F_{i,j}\right)^2}{\sum_{i=1}^{G}\sum_{j=1}^{T+1}(E_{i,j} - Q_{i,j})^2}}$$

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

$$Q_{g,j} = \begin{cases} \begin{cases} 0 \ if \max\limits_{ss=1..SS|tm_{ss}=j} \left(a_{sb_{ss},g} * b_{j,g}\right) * 2 > max\_rating \\ max\_rating \ Otherwise \end{cases} & if \ j \leq T \\ \begin{cases} 0 \ if \ D_g * 2 > T \\ T^2 \ Otherwise \end{cases} & if \ j = T+1 \\ 0 \ otherwise & \end{cases}$$

3.2.3. *Algorithm Operations*

Denote:

- U represents the size of the population.
- $P^e = \{p_i^e \mid i = 1..U\}$ 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. ${p_i^e}_{n,m}$ denote 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

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 $k \mid k = 1 \ldots T$, 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 column to chromosome matrix, apply repair operator to ensure the created chromosome respects constraints $(H1), (H2)$, and $(H3)$.

2. Selection: we implemented the selection process based on Tournament Selection [32]. randomly select φ individuals from P^e and perform a tournament that return the best individuals base on fitness value amongst them.

3. Crossover: Denote p_{father}^e and p_{mother}^e are 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) [33] is one of the most effective crossover technique for ordered list, so it is chosen in this study.

4. Mutation: For each individual p_i^e, have rate B to swap only once for any 2 elements in any column. Similar to generating initial population process, the created chromosome after performing crossover and mutation must be applied 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' satisfy 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's very easy to convert them into the maximum matching problem in bipartite graph. In this study, we use Hopcroft-Karp algorithm [34], a polynomial algorithm to find the maximum matching.

The repair process is performed in 3 steps as follows:

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 $gt \in \mathcal{X}$ (gt 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 gt to h if and only if $t = t_h$, $a_{s_h,g} > 0$ and $b_{t,g} > 0$.

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 aim to retain the good genes from p.

3. Apply the Hopcroft-Karp algorithm [34] to graph $\mathcal{G}$ built in step 1 with pre matching in step 2, we get the final matching which represents for repaired chromosome p'.

4. Experiment and Result

To evaluate the proposed model and algorithm, we use the data obtained 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. 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 received the values in the range of $a_{s,g} \in [0 \dots 5]$ and $b_{t,g} \in [0 \dots 5]$. All experiments mentioned in this report use a computer configured as follows: Processor: Intel(R) Xeon(R) CPU X5650 @2.67GHz (4 CPUs), ~2.3GHz; Memory: 8096MB RAM; all code implemented in java 8.
The pod function is implemented as:

Function: pod
Input: $\{x_{h,g} \mid h = 1..H\}$

$\beth = 100$
r=$NumPod(\{x_{h,g} \mid h = 1..H\})$
$$n = \sum_{h=1}^{H} x_{h,g}$$
If $(1 \le n \le 3)$ and $(r = 1)$ **Return** $\beth$
If $(1 \le n \le 3)$ and $(r = 2)$ **Return** $\beth/5$
If $(1 \le n \le 3)$ and $(r \ge 3)$ **Return** 0
If $(4 \le n \le 6)$ and $(r = 2)$ **Return** $\beth$
If $(4 \le n \le 6)$ and $(r = 3)$ **Return** $\beth/5$
If $(4 \le n \le 6)$ and $(r = 4)$ **Return** 0
If $(1 \le n \le 3)$ and $(r = 1)$ **Return** $\beth$
If $(7 \le n \le 8)$ and $(r = 3)$ **Return** $\beth$
If $(7 \le n \le 8)$ and $(r = 4)$ **Return** $\beth/2$
If $(9 \le n \le 10)$ **Return** $\beth$

The $NumPod$ function defined as:

Function: $NumPod$
Input: $\{x_{h,g} \mid h = 1..H\}$

$Num = 0$
If $(\sum_{t_h \in \{M1,M2,M3\}} x_{h,g} \geq 1)$ **Then** $Num = Num + 1$
If $(\sum_{t_h \in \{E1,E2,E3\}} x_{h,g} \geq 1)$ **Then** $Num = Num + 1$
If $(\sum_{t_h \in \{M4,M5\}} x_{h,g} \geq 1)$ **Then** $Num = Num + 1$
If $(\sum_{t_h \in \{E4,E5\}} x_{h,g} \geq 1)$ **Then** $Num = Num + 1$
Return Num

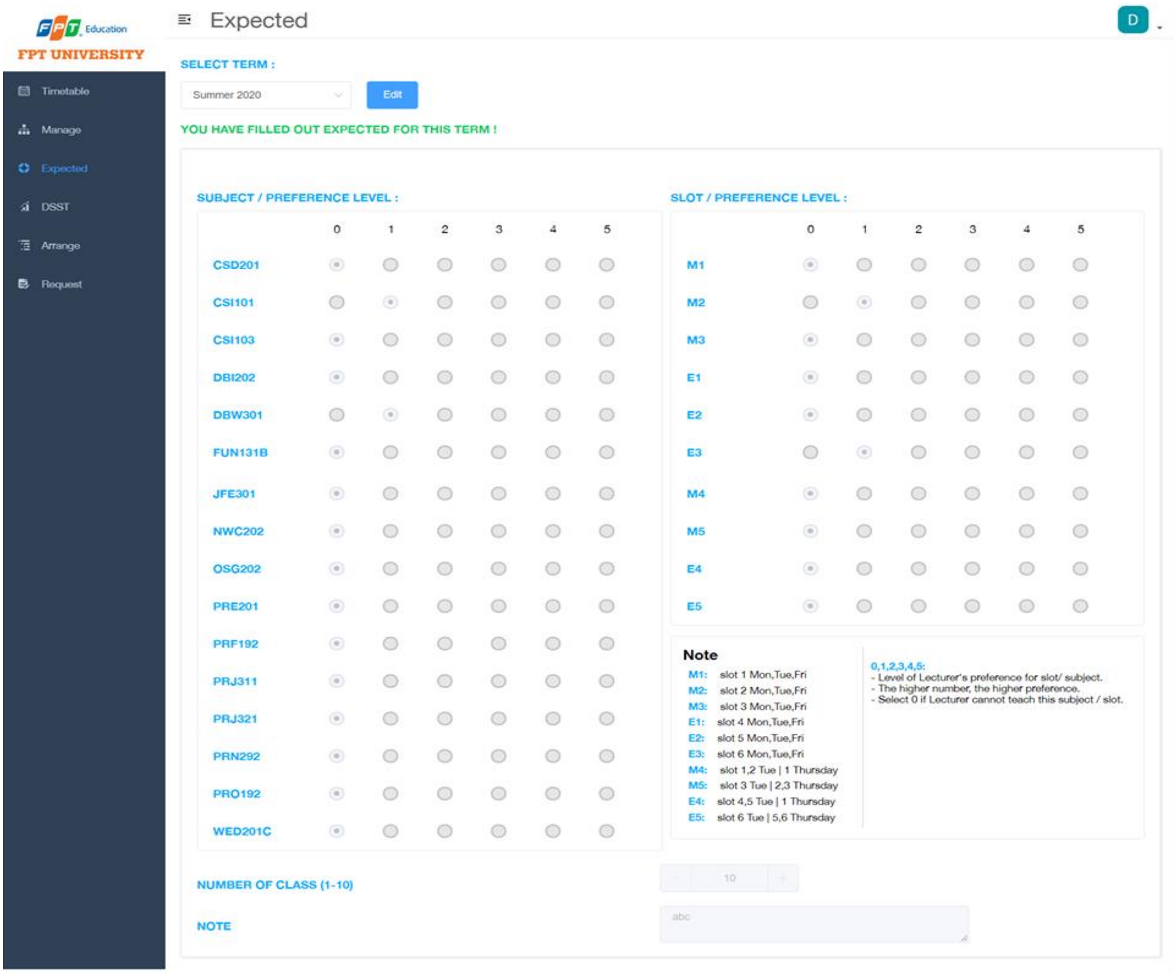

Figure 3: webpage to collect the preferences of a particular lecturer on the subjects and time-slots

Genetic algorithms operate 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.

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

The change of mutation rate value does not affect the convergence result and processing time of the algorithm for both approaches. The tested ranges of the values of the parameters show good results. The small tournament size makes the crossover lose 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 processing time even though it maintains good 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.

Table 3: Parameters to run the algorithm corresponding to approaches

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. Figures 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 seconds, as shown in figure 5. 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 has come very close to the convergence value.

Figure 4: Fitness values of GA over several executions.

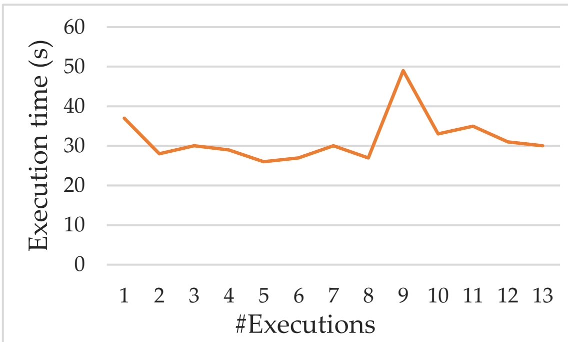

Figure 5: Execution time of GA over several executions.

The proposed model allows faculty preferences for both professional and time needs. 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 level was obtained by executing GA for compromise programming, as shown in figure 7. It observed that teachers who are registered to teach many subjects could teach in many different time frames and naturally prioritize various topics. Meanwhile, with the current scoring of the target function: 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.

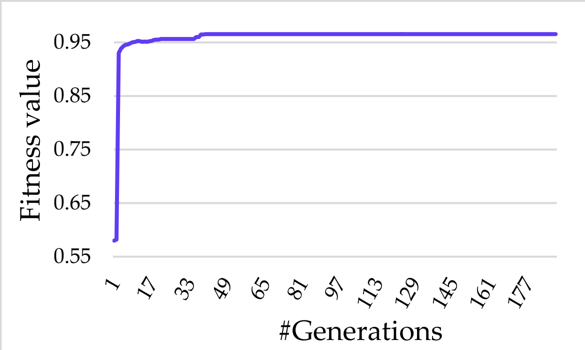

Figure 6: Fitness values changing over generations

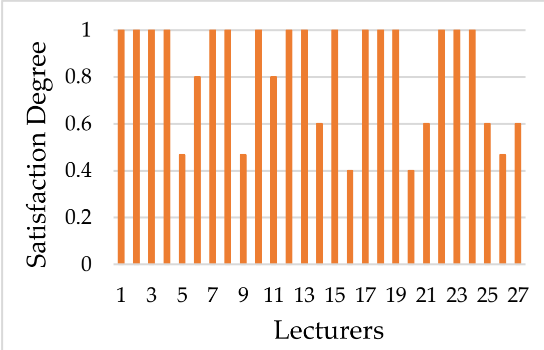

Figure 7: Satisfaction degrees of different lecturers

The Genetic Algorithm (and also other approximation algorithms) does not guarantee to find the global solution. The obtained solutions may be local optima. 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 8. 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.

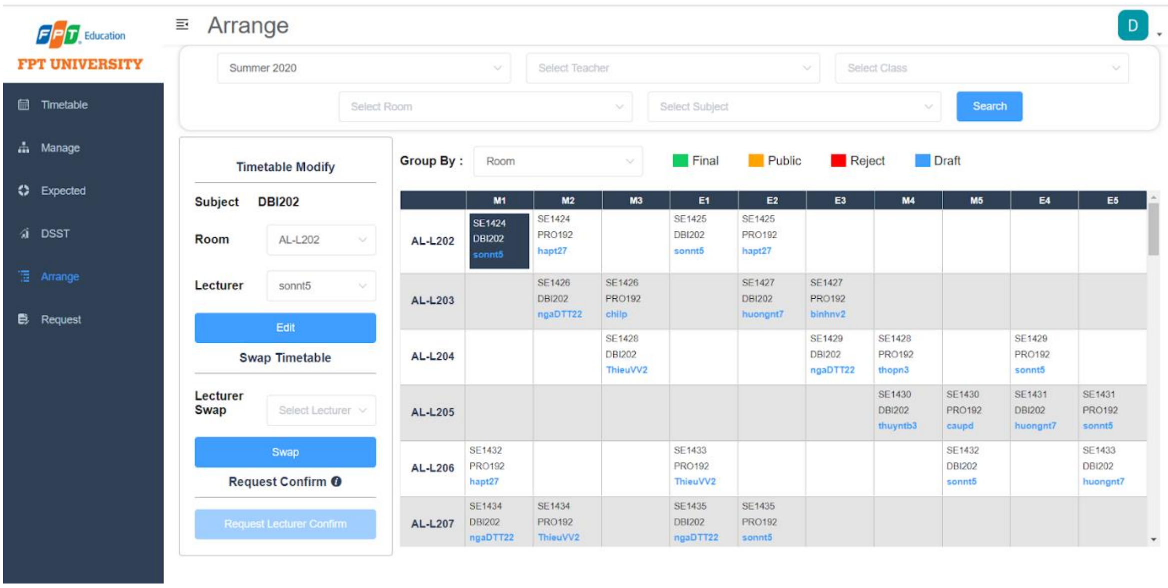

Figure 8: The webpage allows the decision-maker to customize the generated schedule.

5. Conclusion

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 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 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 to 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.

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

References

1. Andrade, P. R. L., Steiner, M. T. A., & Góes, A. R. T. (2019). Optimization in timetabling in schools using a mathematical model, local search and Iterated Local Search procedures. Gestão & Produção, 26(4), e3421. https://doi.org/10.1590/0104-530X3241-19.
2. Lemos, A., Melo, F. S., Monteiro, P. T., & Lynce, I. (2018). Room usage optimization in timetabling: A case study at Universidade de Lisboa. Operations Research Perspectives, 100092. doi: 10.1016/j.orp.2018.100092.
3. Ghiani, G., Manni, E., & Romano, A. (2017). Training offer selection and course timetabling for remedial education. Computers & Industrial Engineering, 111, 282–288. doi: 10.1016/j.cie.2017.07.034.
4. Vermuyten H, Lemmens S, Marques I, Beliën J. Developing compact course timetables with optimized student flows. Eur J Oper Res 2016;251(2):651–61. https://doi.org/10.1016/j.ejor.2015.11.028.
5. Babaei, H., Karimpour, J., & Hadidi, A. (2015). A survey of approaches for university course timetabling problem. Computers & Industrial Engineering, 86, 43–59. doi: 10.1016/j.cie.2014.11.010.
6. D. W. Pentico, "Assignment problems: A golden anniversary survey, "European Journal of Operational Research, vol. 176, no. 2, pp. 774–793, January 2007.
7. Daskalaki S., Birbas T., and Housos E. (2004) An integer programming formulation for a case study in university timetabling. European Journal of Operational Research, vol. 153, issue 1, pp. 117-135.
8. Yang, S., & Jat, S. N. (2011). Genetic algorithms with guided and local search strategies for university course timetabling. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 41 , 93–106.
9. Hmer, A., & Mouhoub, M. (2010). Teaching Assignment Problem Solver. Lecture Notes in Computer Science, 298–307. doi:10.1007/978-3-642-13025-0_32.
10. H. W. Kuhn, "The Hungarian method for the assignment problem," Naval Research Logistics Quarterly, vol. 2, pp. 83–97, 1955.
11. M. L. Fisher, R. Jaikumar, and L. N. V. Wassenhove, "A multiplier adjustment method for the generalized assignment problem," Management Science, vol. 32, no. 9, pp. 1095–1103, September 1986.
12. Lewis R. (2007) A survey of metaheuristic-based techniques for University Timetabling problems. OR Spectrum, vol. 30, issue 1, pp. 167-190.
13. Muthuraman, S., & Venkatesan, V. P. (2017). A Comprehensive Study on Hybrid Meta-Heuristic Approaches Used for Solving Combinatorial Optimization Problems. 2017 World.
14. B. Sigl, M. Golub and V. Mornar, "Solving timetable scheduling problem using genetic algorithms," Proceedings of the 25th International Conference on Information Technology Interfaces, 2003. ITI 2003., Cavtat, Croatia, 2003, pp. 519-524.
15. A. Savić, D. Tošić, M. Marić, and J. Kratica, "Genetic algorithm approach for solving the task assignment problem," Serdica Journal of Computing, vol. 2, pp. 267–276, 2008.
16. V. Sapru, K. Reddy and B. Sivaselvan, "Time table scheduling using Genetic Algorithms employing guided mutation," 2010 IEEE International Conference on Computational Intelligence and Computing Research, Coimbatore, 2010, pp. 1-4.
17. Feng, X., Lee, Y., & Moon, I. (2016). An integer program and a hybrid genetic algorithm for the university timetabling problem. Optimization Methods and Software, (pp. 1–25). doi:10.1080/10556788.2016.1233970.
18. Corne D, Ross P, Fang H (1995) Evolving timetables. In: Lance C. Chambers (ed) The practical handbook of genetic algorithms, vol 1. CRC, Florida, pp 219–276.
19. Nouri, H. E., & Driss, O. B. (2016). MATP: A Multi-agent Model for the University Timetabling Problem. Software Engineering Perspectives and Application in Intelligent Systems, 11–22. doi:10.1007/978-3-319-33622-0_2.

20. Nouri, H. E., & Driss, O. B. (2013). Distributed model for university course timetabling problem. 2013 International Conference on Computer Applications Technology (ICCAT). doi:10.1109/iccat.2013.6521990.
21. Malik, B. B., & Nordin, S. Z. (2018). Mathematical model for timetabling problem in maximizing the preference level. doi:10.1063/1.5041568.
22. Santos HG, Uchoa E, Ochi LS, Maculan N. Strong bounds with cut and column generation for class-teacher timetabling. Ann Oper Res 2012;194(1):399–412. https://doi.org/10. 1007/s10479-010-0709-y.
23. Dorneles ÁP, de Araújo OCB, Buriol LS. A fix-and-optimize heuristic for the high school timetabling problem. Comput Oper Res 2014;52:29–38. https://doi.org/10.1016/j.cor.2014.06.023.
24. David A. Van Veldhuizen , Gary B. Lamont, “Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art” (2000), DOI: 10.1162/106365600568158.
25. 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.
26. Zeleny, M. (1973), "Compromise Programming", in Cochrane, J.L.; Zeleny, M. (eds.), Multiple Criteria Decision Making, University of South Carolina Press, Columbia, pp. 262–301.
27. Ngo Tung Son, Le Van Thanh, Tran Binh Duong, and Bui Ngoc Anh. 2018. 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, 133-138. DOI: https://doi.org/10.1145/3277139.3277149.
28. Ngo Tung Son, Tran Thi Thuy, Bui Ngoc Anh, and Tran Van Dinh. 2019. 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, 125-129. DOI: https://doi.org/10.1145/3316615.3316645.
29. 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.
30. S. M. Thede, “An Introduction to Genetic Algorithms,” Journal of Computing Sciences in Colleges, vol. 20, no. 1, pp. 1–1, 2004.
31. Miller, Brad; Goldberg, David (1995). "Genetic Algorithms, Tournament Selection, and the Effects of Noise" (PDF). Complex Systems. 9: 193–212.
32. Otman, A., & Jaafar,A. (2011). A comparative study of adaptive crossover operators for genetic algorithms to resolve the travelling salesman problem. International Journal of Computer Applications, 31(11), 49-57.
33. John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM Journal on Computing, 2(4):225–231, 1973.