www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 12(3), Sep-Dec 2021, 32-49. ISSN: 2007-1558.

Personal Course Timetabling for University Students based on Genetic Algorithm

Brenda Sunuami González López[1]; René Arnulfo García Hernández[2], Yulia Ledeneva[3]
Universidad Autónoma del Estado de México, Unidad Académica Profesional Tianguistenco, Instituto Literario No. 100, CP 50000, Toluca, Edo. Mex, México
[1] brenda_sunuami@hotmail.com
[2] renearnulfo@hotmail.com (corresponding author)
[3] yledeneva@yahoo.com

Abstract. This paper presents the optimization problem of generating a Personal University Course Timetabling (PUCT) for students from a Published Course Catalog (PCC) and an evolutionary approach to solve it. A PPC is a result of a manual or automatic process of a university course timetabling system to assign classes to times and spaces, considering constraints and preferences. However, independently of how the PPC was created, each university student faces particular constraints to select its timetable's courses. Due to the complexity of generating the timetable manually, several students were currently not considered a better option. In this paper, we present a method based on a genetic algorithm that considers the constraints and preferences presented by students at the Autonomous University of the State of Mexico for generating a PUCT. According to experimentation with a ground truth dataset, the proposed method not only shows better quantitative results than manual generation of the timetable but also shows good qualitative results based on the evaluation of the students.

Keywords: personal course timetabling, multiobjective optimization, combinatory optimization, micro genetic algorithm.

Article Info
Received Jan 19, 2021
Accepted May 11, 2021

1 Introduction

Nowadays, the covid-19 pandemic has accelerated the paradigm shift occurring in higher education institutions (referred to as universities and colleges) because the traditional and conventional face-to-face education has abrupt change to distance education. In traditional education, the concept of a student is considered a full-time student that must follow a fixed curriculum scheme with a fixed course timetabling. However, now the curriculum of higher education institutions is more flexible with a modular approach based on the student learning profile to adapt to the demands of the changing world. Now, it is expected that a university student could course complementary courses in other campuses, universities, or countries. In the new central-student learning, it needed to recognize that the students have time constraints and preferences to find a personal course timetabling within the limits of the curriculum. Curriculum flexibility and distance education have generated a greater offering of courses, seminars, and workshops increasing the complexity involved in generating the personal timetable manually for each student.

According to some Software Engineering students at the *Universidad Autónoma del Estado de México* (UAEMEX), finding a good *Personal University Course Timetabling* (PUCT) is not a trivial task that affects their performance. For example, UAEMEX students mentioned that they were unsure if their current PUCT was the best option or could be improved. Also, the students mentioned the problem of selecting the best PUCT consists in finding a set of available courses that considers the current progress of approved courses but also the failed courses, the available hours and days, the total minimum and maximum credits by period, the overlap of schedules, and the gap in dead hours between courses.

This paper addresses the optimization problem of generating a PUCT for university students from the university courses offered from a published catalog. It is worth mentioning that a university course catalog is a result of a human-made process of a well-studied university-course timetabling system to assign classes to times and spaces considering institutional limitations (such as classrooms, labs, and instructors) and, in general, students and instructors' preferences. In our review, most of the related work about university course timetabling is related to the optimization problem of generating a university course timetabling. Müller

[24] defines it as: "a resource allocation problem with the aim of assigning classes to times and spaces in such a way that no two classes are placed in the same room or taught by the same instructor at any given time". Additional preferences and constraints should be considered, such as: time, room, and distribution preferences; and student conflicts. The variety of preferences and constraints, the diversity of the problem, and the specific requirements make the problem more complex [1]. The timetabling problem is a so-called NP-hard and NP-complete optimization problem, depending on the constraints [2].

Therefore, the PUCT problem of this paper is different because it is a consequence of the result of the traditional university course timetabling systems focused more on the student than on the institution. However, regardless of how the university course catalog was created, each student faces particular limitations in selecting courses to their timetable, and due to the large search space involved in finding an optimized schedule, some students currently did not consider a better timetable because they did not try all the options. In specific, in this paper, we present a method based on an evolutionary algorithm that considers six constraints that present the UAEMEX students for generating a PUCT; however, the proposed method could easily be adapted to other universities.

To obtain not only a quantitative evaluation but also a qualitative evaluation of the proposed method, a ground truth dataset with UAEMEX students was collected. In particular, for the qualitative evaluation, the currently manually made PUCT of the UAEMEX students were compared to those proposed automatically.

This paper is organized as follow: a brief review of the related work about the traditional university course timetabling problem and its solution methods are mentioned in section two, a description of the primary and second constraints of the Personal University Course Timetabling (PUCT) problem for UAEMEX students is described in section three, a detail explanation of the proposed method is presented in section four, a description of the dataset and the discussion of the experimentation are presented in section five, and the conclusions are giving in section six.

2 Related work

Timetabling problems have used different optimization techniques based on swarm optimization, evolutionary and local-search algorithms, see Fig. 1, which are considered metaheuristic algorithms. Evolutionary computing generationally produces solutions to an optimization problem with the characteristic that in each generation, the results usually improve on the previous generation [3], for example, Greedy Algorithm[4], Genetic Algorithms [5], Memetic Algorithms[6], and Simulated Annealing [7].

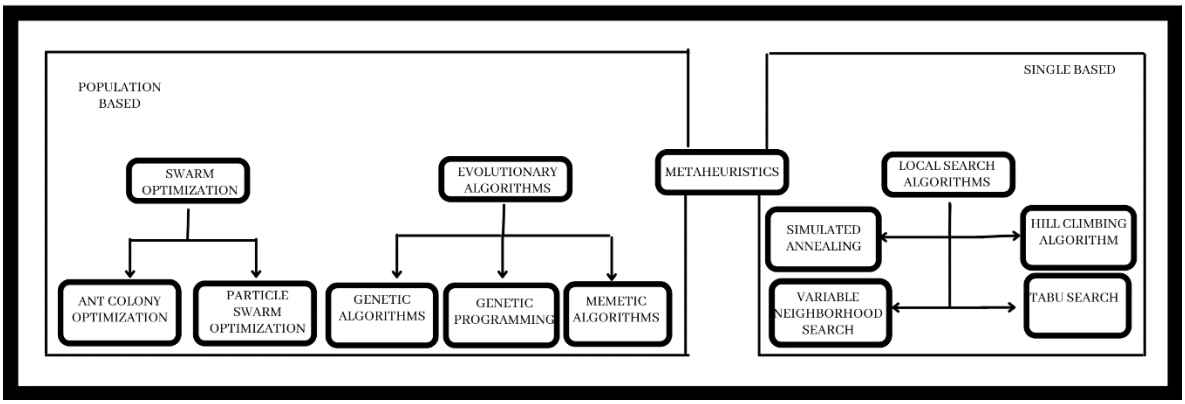

Fig. 1 Metaheuristic algorithms.

Timetabling optimization is divided into several sections. For example, *transport timetabling* solves the route assignment problem to drivers; *sports timetabling*, specifically in soccer, has the characteristic that there are different tournaments between teams such as all-against-all that requires a specific type of programming; *employee timetabling and roistering* for the assignment of shifts in the health sector; and *educational timetabling* for programming the related tasks to the schools and higher educational institutions [8]. For example, *exam timetabling* where an assignment of defined timetabling is carried out since there are flexible timetabling for the corresponding exams implemented in a university [2], school and higher education institutions timetabling to assists to the distribution of times, rooms, courses and instructors preferences; and student conflicts. However, managing student preferences

and conflicts is about the overlapping course that students present in general, not for each student. The only system that is more related to our concern is the next.

Most of the timetabling research is focused on high schools and universities due to the flexible hours (flexibility to choose the courses to take and the most convenient hours). In [9], a complete course timetabling and student timetabling system developed for the University of Waterloo between 1979 and 1985 is described. The system is based on a "demand-driven" philosophy, where students first chose their courses, and the system tries to find the best timetables to maximize the number of satisfied requests. The problem is first decomposed into small manageable subproblems. Each subproblem is solved in sequence using a greedy heuristic to assign times to sections and a Lagrangian relaxation algorithm to assign classrooms. Representatives from each department have interactive access to make modifications. Finally, each student is assigned individually to the combination of sections that maximize the satisfaction of the schedule and the classroom section. The system was used successfully for 15 years.

An example of higher education timetabling of course and room assignment for the engineering faculty of the Diego Portales University where the problem is that in each semester there is an average of 150 courses which have a variable number of sections [10]. In this case, the courses have two types of classes: chair and auxiliary. At the beginning, the timetabling was generated by a team made up of three professionals, who took an average of one month to obtain the final timetabling. This research is composed of soft and hard constraints (where soft constraints are those that are a possibility of restrictions to be carried out, on the other hand, hard constraints are those that must be complied with). Some soft constraints are: the auxiliary classes should preferably be held on Wednesdays in any hour block, but if this assignment is not possible, it can be done on any day and time block. Avoid, as far as possible, assigning courses to the auditorium. Some hard constraint is: each course must be assigned to a classroom with a capacity for students for that course, an instructor cannot teach more than one class at a time, the available-schedules of the instructors must be respected, and there should not be time limits between courses in the same semester.

In [11], the use of a simulated annealing algorithm is described to solve the problem of school timetables, and it is compared to the basic geometric-cooling program (a scheme that uses two cooling speeds and four schemes) that allow both reheating and cooling. Schematics are variants of the basic geometric cooling program. The found is that using multiple cooling speeds is more effective than one. Four schemes were experimented with raising the temperature to allow the system to escape local minima. In general, the scheme using the phase transition temperature in combination with the best quality of the solution found produced the best results. In this work, the author only tested the schemas with the timetabling problem.

In [12], the design and implementation of a PC-based computer system to assist in the construction of a combined timetable of college courses and exams is described. The system uses an integer programming model that assigns courses, schedules, and rooms. The model is combined with a flexible front-end device that generates constraints corresponding to the conditions specified by the user and report writers that facilitate the presentation of the resulting timetable. The quality of the developed timetabling depends on the relative position of the courses assigned to the available time periods, a condition that the integer programming model attempts to satisfy by constructing groups of courses that are assigned to groups of time periods. In addition, the objective function is used in a way that takes advantage of the experience and knowledge of the user about the problem. The solution to the course timetabling problem is used to build an initial solution to the exam schedule. A heuristic algorithm is generated to further improve it until a good workable solution is reached. The whole system is flexible and allows the easy construction and testing of alternative programs that are predefined according to user-specified requirements. The Athens University of Economics and Business has used the system with success.

The author [13] mentions that the article has been motivated by a real-life class-teacher schedule problem, where there are several classes, and each class follows its own fixed curriculum. A set of available working days is provided within which all class syllabi must be completed. Each curriculum consists of a specific set of subjects. For each subject, you specify the first workday (published date) on which it can start and the last workday (end-date) on which it must be completed. In agreement with the author Hertz, a new approach to finding a viable course timetabling was developed. The model can handle timetabling problems with selected courses, time windows, and pre-assignment requirements, compactness, geographic limitations, and precedence. Also, in their initial requirements, teachers do not necessarily specify the duration of each course. In most of the approaches described in the literature for tackling course timetabling problems, these lengths are assumed to be initially set to find a feasible course timetable by planning taboo search. No such restrictions are imposed that can drastically reduce the size of the feasible course program set. Finally, the method developed in this article can be easily implemented by using object-oriented programming.

3 Problem description

Usually, a university publishes the offered courses (also: seminars, lectures, and workshops) of the curriculum to be run in the next period in a catalog where days, hours, classroom, labs, and instructors of the courses are specified. The information, rules, and the suggestions to select the courses are specified in the curriculum, for example, at the UAEMEX for each course is specified the number of credits, the mandatory or desirable sequence of the courses, the mandatory and elective courses, the number of maximum and minimum credits that a student can course in a period, and the number of the desirable period completed. University course catalog is the result of a human-made process or of a well-studied university-course timetabling system to assign classes to times and spaces considering institutional limitations (such as classrooms, labs and instructors) and, in general, students and instructors preferences.

Given a published catalog of available university courses that a student can select, *the Personal University Course Timetabling (PUCT) problem* consists in finding the set of courses that most benefits the student according to the limits of curriculum flexibility, the current progress within the curriculum, and the hard and soft constraints about time and curriculum suggestions.

3.1 Time

Even though the timetabling can be made in different period times, day, week, weekend, bi-week, and month; in this paper, by the UAEMEX curriculum, the timetabling is constructed by week from Monday to Saturday from 7am to 7pm; and it is repeated every week during a semester. The number of hours of a course by week varies from three to five hours distributed in the week where the minimum time for a meeting is one hour, and there is no maximum. For example, a three-hour course is distributed in one meeting (3h course) or two meetings of one and two hours (1+2h course) and vice versa (2+1h course), or two meetings of one and a half hours (1.5+1.5h course). Usually, the courses of three and four hours are divided in two meetings and the courses of five hours are divided in two or three meetings; with a day as a gap between meetings. Classes should generally end 10 or 15 minutes early so that students and the instructor can move from one class to another. Two courses have time overlap if and only if some time is shared in at least one meeting on the same day. According to the UAEMEX students, they have two-time constraints:

Overlapping Class Time (OCT) is a hard constraint because a student cannot be in two places at the same time, and it decreases the academic performance of the student. Even though it is evident, we find some cases with one-hour overlap and half-hour overlap.

Maximum Weekly Time at the University (MWTU) is a hard constraint because some students have other activities like jobs, sports, and other courses. Students are mentioning that the other activities are flexible, but they need to complete a time. For this, first, they give preference to the university time, and then they define other timetabling. The MWTU is the sum of the hours per day from the first-hour course until the last-hour course. Therefore, all the free hours between classes in a day also are considered per day. The MWTU constraint minimizes the number of dead hours between classes. In this sense, the compactness of the timetabling can be maintained by the MWTU.

It is worth mentioning that a system requirement, beyond the PUCT problem, is to consider that from the moment the course catalog is published until the student makes the registration, the published catalog may change over time since some courses are closed because the classroom capacity is filled or new courses are opened at the last minute.

3.2 Curriculum

The information, rules, and the suggestions to select the courses are specified in the curriculum. In traditional education, the concept of student is considered a full-time student that must follow a fixed curriculum scheme with a fixed course timetabling. However, now the curriculum of higher education institutions is more flexible with a modular approach based on the student learning profile to adapt to the demands of the changing world. Curricular flexibility is based on the credits of the courses, which is a unit of measurement of academic work that makes it possible to find equivalence between different subjects. In this sense, the credits of a course are directly related to the weekly class meeting hours. In each semester period, a set of courses are suggested; however, each student can select later or earlier courses according to the mandatory and elective sequence of courses. Therefore, each course has a number indicating the suggested period. According to the curriculum, there are two constraints:

Maximum-Registered Credits (MAXRC) is a hard curricular constraint to limit the maximum number of credits that a student can register in a period.

Minimum-Registered Credits (MINRC) is a hard curricular constraint to limit the minimum number of credits that a student can register in a period. Of course, only in the last period is it possible to get less MINRC.

Suggested Courses (SC) is a soft constraint according to the current progress of the student in the curriculum that suggests the sequential relevance of the course for the timetabling using the suggested period of the course. The idea is that an early-period course should be completed before a later course, but there is flexibility.

To-Advance Courses (TAC) is a soft constraint to indicate the maximum number of courses that a student desires to advance beyond the current progress of the student.

Given that there are many possibilities of timetabling for each set of personalized conditions of the student, then this is an optimization problem, which must be evaluated according to the conditions that adapt to the conditions of the student, how to find the best workload that considers the student's current progress, student preferences, and curriculum restrictions.

4 Proposed genetic algorithm

In this paper, an evolutionary approach to the PUCT problem is proposed. The hypothesis is that an evolutionary approach can find better timetables according to the student preferences and curriculum constraints. In other words, we treat this problem as an optimization problem using an evolutionary approach.

The input of the proposed genetic algorithm is the published university course catalog, curriculum constraints, student conditions and preferences; as is shown in Fig. 2. After the GA search, the PUCT is obtained. Genetic Algorithm (GA) is an evolutionary approach inspired in the natural selection theory proposed by Charles Darwin (Darwin, 1859) that has proved to be an alternative solution for global optimization problems in large search spaces [44–48]. In the first step, the GA uses a population of random solutions (*initial population step*) evaluated according to the objective function to optimize (*fitness function step*). In evolutionary optimization, a solution for one problem is not absolute; it means there is a set of possible solutions where some are better than others. Mainly considering the best solutions (*parents selection step*), the GA proposes a new population mixing (*crossover step*) some parts from a canonical codification (*chromosome encoding step*) from good solutions in order to get better solutions (*evolution principle*). Eventually, the way of mixing some parts from the canonical codification could produce repeated solutions. Therefore, the GA applies a small variation (*mutation step*) to the canonical codification in the new population in order to explore new solutions. The new population is evaluated, and the process is repeated until a satisfactory solution is reached or until some arbitrary stop-criteria is reached (*stop condition*).

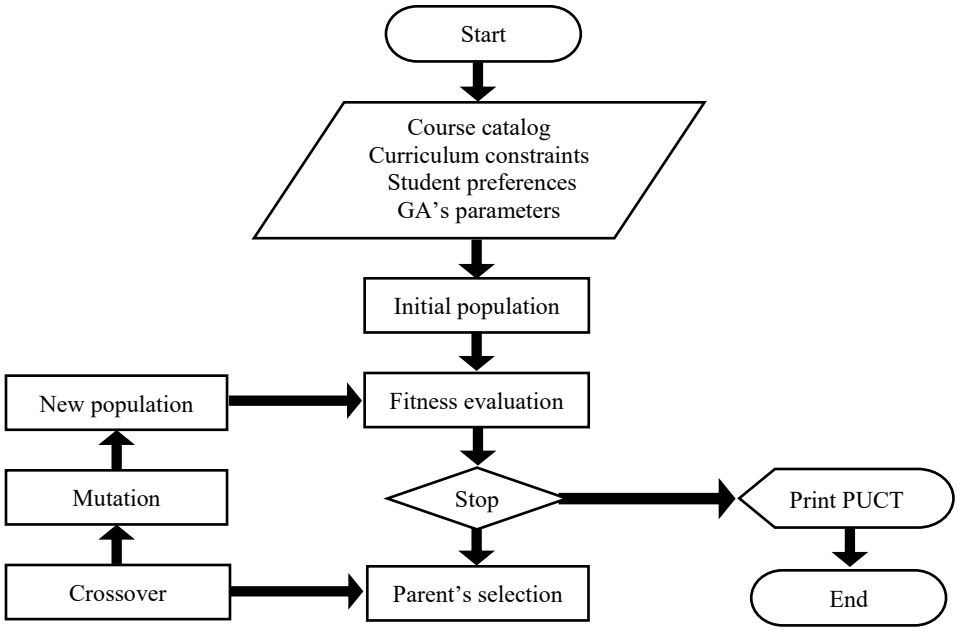

Fig. 2 Flow chart of proposed genetic algorithm.

4 .1 Proposed genetic operators

The fitness function (F) evaluates the PUCT represented by the chromosome (t) as a multiobjective function, see ec. (1). The fitness function considers the curriculum constraints: minimum curricular credits (Y) and maximum curricular credits (X); and the student preferences: Maximum Available Weekly Hours at the University (M) and student-defined maximum credits ($K \leq X$). The constraints measure the error in a range of 0 to 1, therefore F must be minimized. The function is formulated as the weighted sum of the individual evaluations. The variables $\alpha, \beta, \gamma, \varepsilon, \lambda, \psi, \theta$, and Ω; are used to students may give a greater weight to certain constraints, where hard constraints cannot be cero and must to receive a greater or equal weight than soft constraints. The sum of the conditions must be 1. The fitness function for the genetic algorithm is:

$$F(t) = \alpha\, OCT(t) + \beta\, MWTU(t) + \gamma\, SC(t) + \psi\, MAXRC(t) + \phi\, MINRC(t) + \Omega\, TAC(t)\,. \quad (1)$$

The constraints are:

1. *Overlapping Class Time* (*OCT*). OCT hard constraint of the timetabling t is evaluated considering the *Number of Overlapping Hours* (*NOH*) in all t divided by M that the student specified, as ec. (2).

$$OCT(t) = \frac{NOH(t)}{M}\,. \quad (2)$$

2. *Maximum Weekly Time at the University* (*MWTU*). The MWTU constraint of t is evaluated with the *Number of Extra Hours* (NEH) of t that exceeds to M:

$$MWTU(t) = \begin{cases} \frac{NEH(t)-M}{M}, & NEH(t) > M \\ 0 \quad , & in\ other\ case \end{cases}. \quad (3)$$

3. *Suggested Courses* (*SC*). SC is a soft constraint that should suggests the assignment of earlier courses before the later ones. In a flexible curriculum, a student can advance or delay courses therefore the selection of the courses of the chromosome t should include the courses from earlier semesters (weight to minimize the SC constraint) that the student can take. For this, the semester period of the course suggested by the curriculum is used (see Fig. 3). Given the n courses of a timetabling t and the catalog r of the available courses for the student, the suggested courses uses the suggested-semester period of courses ($Course^{period}$) as a weight of the selection. Therefore, the sum of the n periods of the courses from t is calculated in $S(t)$, see ec. (5). To normalize $S(t)$, the sum of the n periods of the earlier courses from r is calculated in $L(t)$ as the smallest weight, see ec. (6), and the sum of the n periods of the later courses from r is calculated in $H(t)$ as the largest weight, see ec. (7). The SC ec. (4) is computed as follow:

$$SC(t) = \frac{S(t)-L(t)}{H(t)}, \quad (4)$$

where:

$$S(t) = \sum_{i=1}^{n=|courses\ \in t|} Course_i^{period \in t}\,, \quad (5)$$

$$L(t) = \sum_{i=1}^{n=|courses\ \in t|} Course_i^{earlier\ \in r}\,, \quad (6)$$

$$H(t) = \sum_{i=1}^{n=|courses \in t|} Course_i^{later \in r} . \quad (7)$$

4. *Maximum-Registered Credits* (*MAXRC*) is a hard curricular constraint to limit the maximum number of credits that a student can register in a period according to the maximum curricular credits (X), see ec. (8).

$$MAXRC(t) = \begin{cases} \frac{NC(t)-X}{X}, & NC(t) > X \\ 0 \quad , & NC\ (t) \leq X \end{cases} . \quad (8)$$

5. *Minimum-Registered Credits* (*MINRC*) is a hard curricular constraint to limit the minimum number of credits that a student can register in a period according to the maximum credits (Y) specified by the curriculum. Of course, only in the last period is it possible to get less MINRC, see ec. (9).

$$MINRC(t) = \begin{cases} \frac{Y-NC(t)}{Y}, & NC(t) < Y \\ 0 \quad , & Y \leq NC(t) \end{cases} . \quad (9)$$

6. *To-Advance Courses* (*TAC*) is a soft curricular constraint to allow more flexibility when the student desires to increase the load academic work, so this constraint is designed to try to complete a *student-defined maximum credits* (K); see ec. (10). Of course, K cannot pass to maximum curricular credits (X).

$$TAC(t) = \frac{\sum_{i=1}^{n=|Courses \in t|} Creditos(Course_i)}{K}, \ k \leq X . \quad (10)$$

Chromosome Encoding. A chromosome (t) represents a PUCT for a student S. The number of genes of t correspond to the number of offered available courses that the only the student S can course which is a selection of the complete published course catalog. The genes of the chromosome t are represented by a binary encoding where 1 means "aggregate this course to the PUCT" and 0 means "disaggregate this course to the PUCT".

Initial Population. All chromosomes in the initial Population (P_0) are created in random way.

Parent Selection. Once each chromosome has an associated fitness value, those stronger chromosomes have more probability of being selected as parents (natural selection mechanism). Natural selection mechanism establishes that two good solutions (chromosomes) could produce better solutions; nevertheless, in some cases the solution could be worse. In this step, the classical μ-tournament selection is employed where the strongest chromosome from a small random subsample (μ) is selected as a parent. The smaller the μ subsample, the higher the promising chromosome review.

Crossover. n-point crossover is used for mixing the genetic information of the parents, where n random points between the genes of the parent chromosomes are selected, and then two offspring chromosomes are created swapping everything between the selected points.

Mutation. According to the evolution scheme, the mutation slightly happens in nature, with a low probability of 0.1 percentage. However, it is one of the fundamental mechanisms to preserve evolution. Since the chromosome is binary, the classical inverse mutation operator is used.

Elite strategy. It helps to keep the best solution from the previous generation to the new generation.

5 Experimentation

In the next section, the features of the ground truth dataset used to the experiments is shown. In particular, the curriculum description, the complete published course catalog, the student dataset description are described. Using this dataset, the experiments design to tuning and testing our proposed method is described.

5.1 Dataset description

Due to there is not a PUCT benchmark dataset, the dataset was collected in an interview with software engineering students at the UAEMEX. For this dataset, the last-period students participate in this dataset because they are experts in the manually-made PUCT process. In the next section, the curriculum description, the complete course catalog, the student conditions, and the student timetabling are described.

Curriculum description is of the software engineering career with 55 courses distributed at nine semesters, but the number of semesters is flexible. The courses set suggested by semester is shown in Fig. 3. For example, the six courses AU11-UA16 at the third semester are suggested as an academic work-load of 35 credits, but each student may select its courses. For example, in the third semester, the UA12 maintains an obligatory sequence with UA7, so it cannot be coursed if UA7 has not been approved. The minimum number of credits by a period is 24 and the maximum number of credits by period is 64. A software engineering student must complete 348 credits to finish the career.

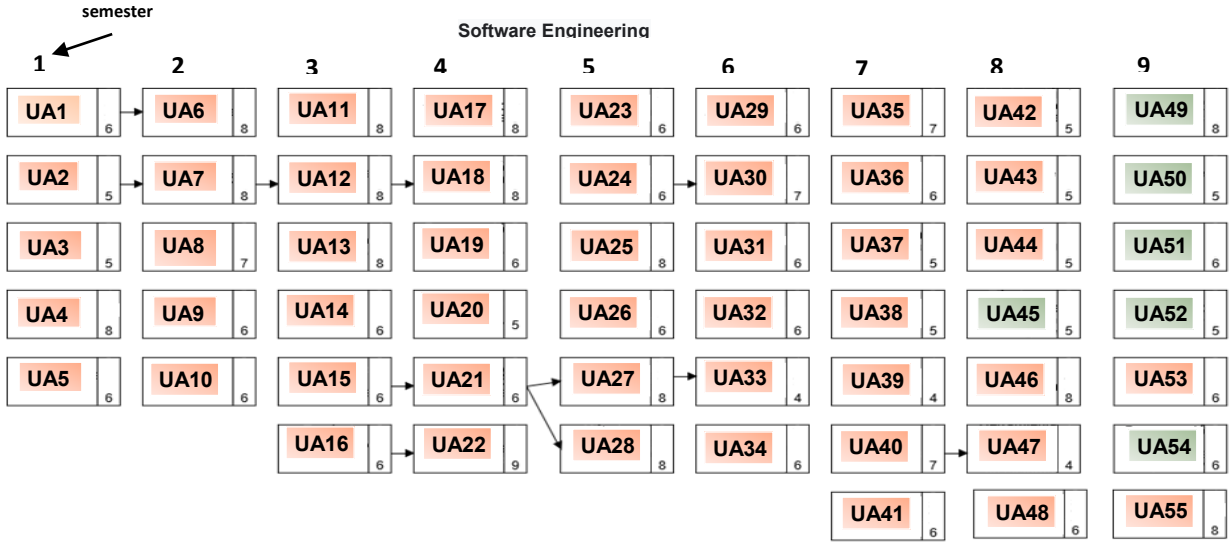

Fig. 3 Distribution of the software engineering curriculum.

Student dataset description. In the dataset, 25 students participate in the experiments, which correspond to a whole last-semester group. This sample was chosen since there were students with different academic performance: 5 students have approved all their courses with the suggested curriculum credits with ideal performance, 5 students have failed courses with a bad performance, and 15 students have approved all their courses, but their total credits is lower than the suggested by the curriculum, so they have slow performance. The idea of this data set is to show that the students have the experience to make PUCTs, and they have different and particular conditions.

Published course catalog. Fig. 4 shows the available course catalog that were used for the experimentation.

Student timetable description. For the collection of data regarding the PUCT of the students, in the first instance, an examination was applied to identify the conditions that the students take into account to carry out the PUCT; as a second instance, it was asked to consider the published course catalog and as a third instance make the PUCT. Table 1 shows the conditions of the students and the evaluation of the PUCT by the fitness function.

Hour/day	M	T	W	T	F	S
7:00 - 8:00		UA51			UA50	
8:00 - 9:00						
9:00 - 10:00						UA54
10:00 - 11:00	UA52		UA49	UA53	UA55	
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						

Hour/day	M	T	W	T	F	S
7:00 - 8:00	UA45	UA46		UA47		
8:00 - 9:00						
9:00 - 10:00			UA44	UA48		
10:00 - 11:00	UA42					
11:00 - 12:00						
12:00 - 13:00		UA43				
13:00 - 14:00						

Hour/day	M	T	W	T	F	S
7:00 - 8:00	UA50		UA52		UA55	
8:00 - 9:00						
9:00 - 10:00						
10:00 - 11:00	UA49	UA51	UA54	UA53		
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						

Hour/day	M	T	W	T	F	S
7:00 - 8:00	UA44	UA45				
8:00 - 9:00						UA42
9:00 - 10:00						
10:00 - 11:00	UA46	UA48				
11:00 - 12:00						UA43
12:00 - 13:00		UA47				
13:00 - 14:00						

Fig. 4 Available course catalog.

Table. 1. Student conditions to generate manual and automatic PUCT for 25 students evaluated with the proposed fitness function

Student	Student conditions						Fitness Evaluation
	OCT	MWTU	SC	MAXRC	MINRC	TAC	Manually-made PUCT
E1	20	0	6	42	17	0	0
E2	15	0	5	42	17	1	0
E3	14	1	6	35	17	0	0
E4	15	2	5	41	17	1	0
E5	16	1	6	42	17	1	0
E6	18	0	6	36	17	0	0.07
E7	20	0	6	42	17	0	0.08
E8	17	1	6	41	17	0	0.02
E9	16	1	6	38	17	0	0.08
E10	20	0	6	42	17	0	0.02
E11	16	1	6	42	17	0	0.08
E12	14	0	6	35	17	0	0.06
E13	20	0	6	34	17	1	0.06
E14	25	0	5	42	17	0	0.08
E15	24	1	5	37	17	0	0.07
E16	22	0	6	42	17	0	0.8
E17	17	1	6	36	17	0	0.02
E18	14	0	5	38	17	1	0.08
E19	20	0	6	42	17	0	0.03
E20	15	0	6	42	17	1	0.06
E21	18	1	6	42	17	0	0.08
E22	20	0	6	38	17	0	0.07
E23	17	1	5	38	17	0	0.08
E24	18	1	5	42	17	0	0.08
E25	20	0	6	38	17	0	0.07

5.2 Parameter Tuning

The following experiments have the objective of showing the evolutionary performance of the proposed genetic algorithm to see if it is possible to obtain good performance with a set of predefined parameters. In all experiments, the next AG parameters are used: 2-tournament selection, 2-point crossover, the population size of 100, and maximum generation number of 100. For the following experiments, the preferences and PUCT of students E1, E2, and E3 were used. In all the experiments, the variables α, β, γ, ε, λ, ψ, θ, and Ω receive the same weight (0.125).

The first parameter to determine was the crossover probability, which was tested as 0.3, 0.6, and 0.9; with a fixed mutation probability of 0.3, see Fig 5. As can be seen, in all cases, the proposed method can generate an adequate PUCT for the student E1, thus the crossover probability 0.3 seems adequate to obtain the better result before the other options. In problems with several local optimum, this may not be desirable because it leads to premature convergence in the search, but in this experiment it does not appear to be the case.

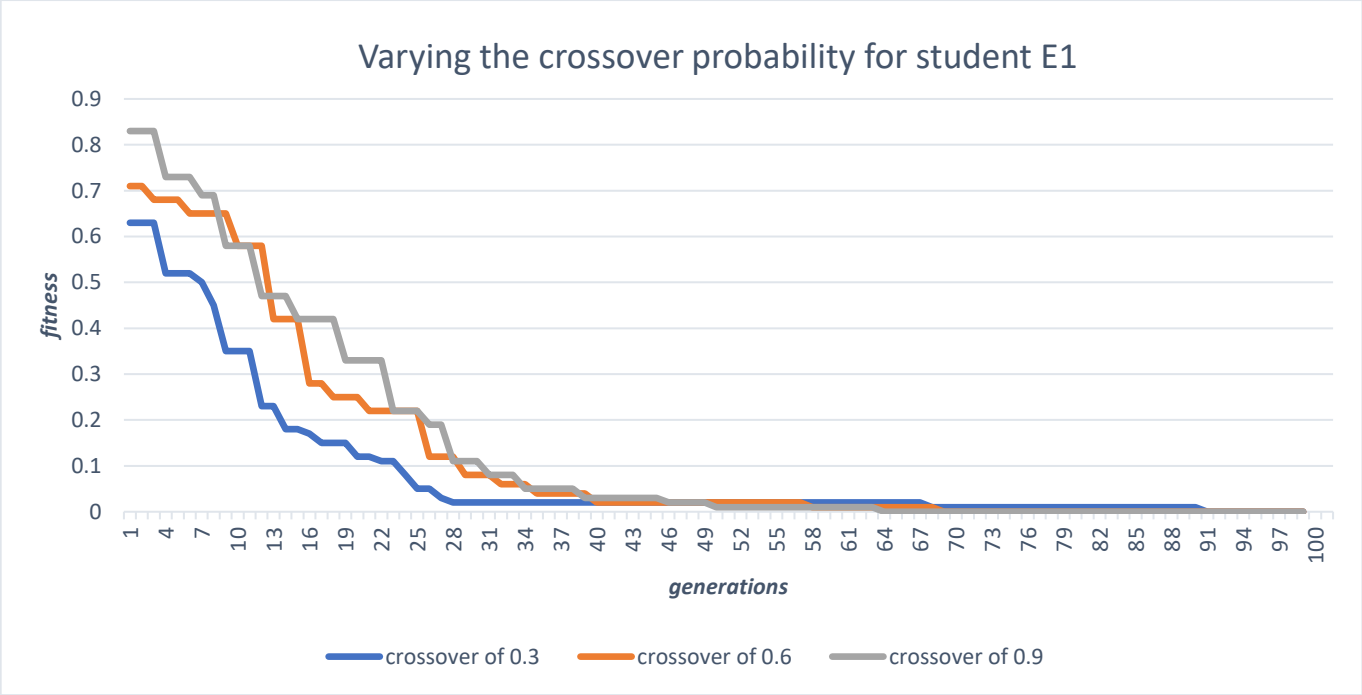

Fig. 5 Varying the crossover probability for student E1.

The second parameter to determine was the mutation probability, which was tested as 0.03, 0.06, and 0.09; with a fixed crossover probability of 0.3, see fig. 6. As can be seen, in all cases, the proposed method can generate an adequate PUCT for student E1, however, the mutation probability of 0.03 seems more appropriate.

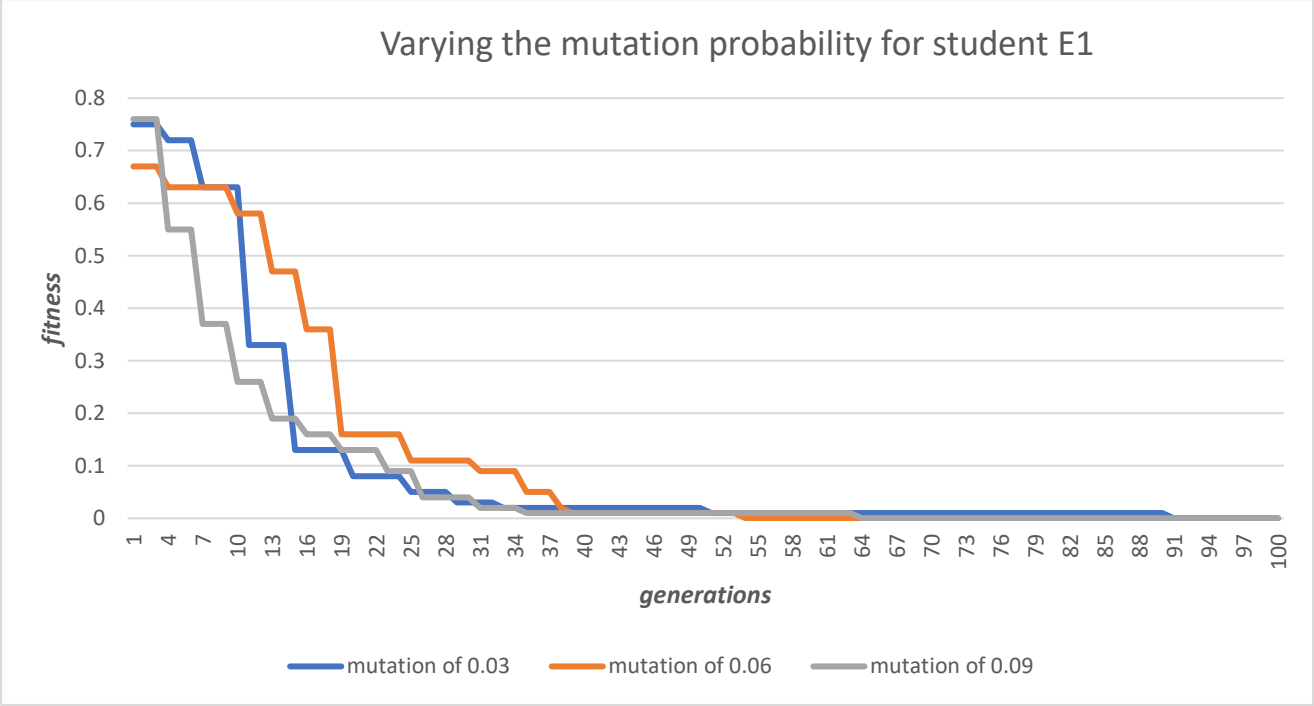

Fig. 6 Varying the mutation probability for student E1.

Fig. 7 shows the performance of the proposed GA using the crossover probability of 0.3 and the mutation probability 0.03. In order to observe if this behavior does not depend on the conditions of the student E1, the same scheme parameters were repeated on students E2 and E3.

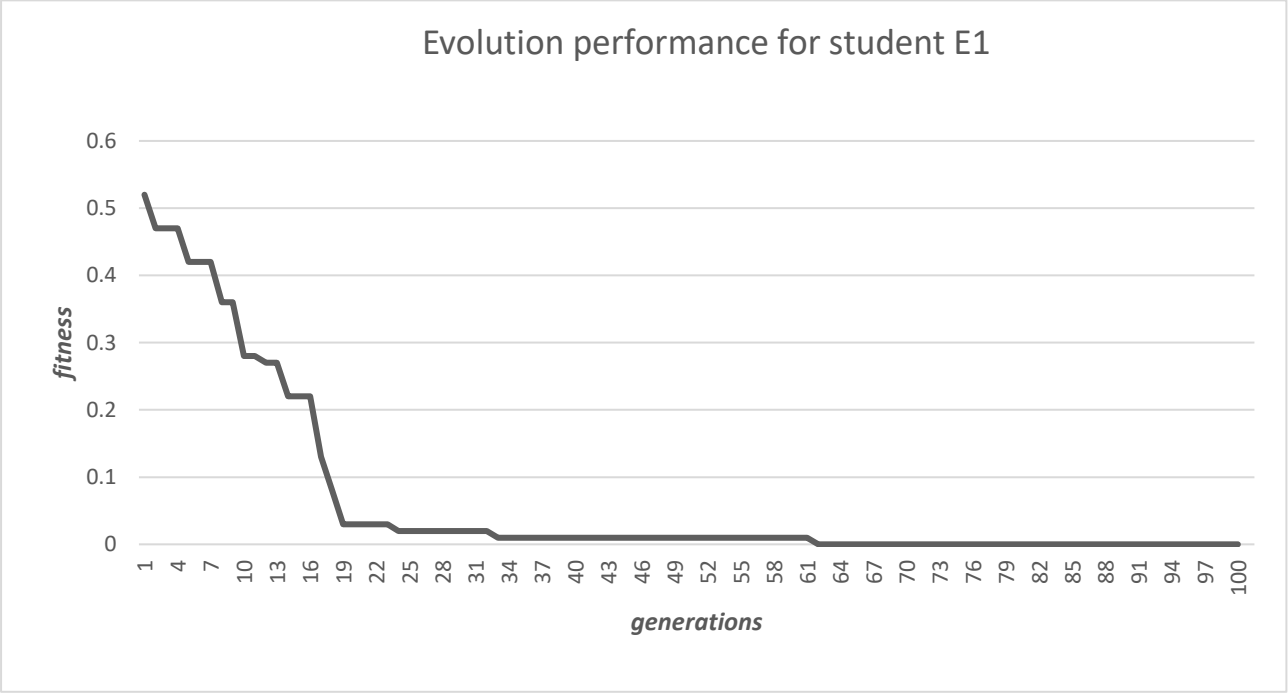

Fig. 7 Evolution performance for student E1.

In Fig 8, the crossover probability ranged from 0.3, until 0.9; with a fixed mutation probability of 0.3. As can be seen in Fig 8, in all three cases, the proposed method can generate an adequate PUCT for the E2 student; but the crossover probability of 0.3 seems adequate to obtain a better result.

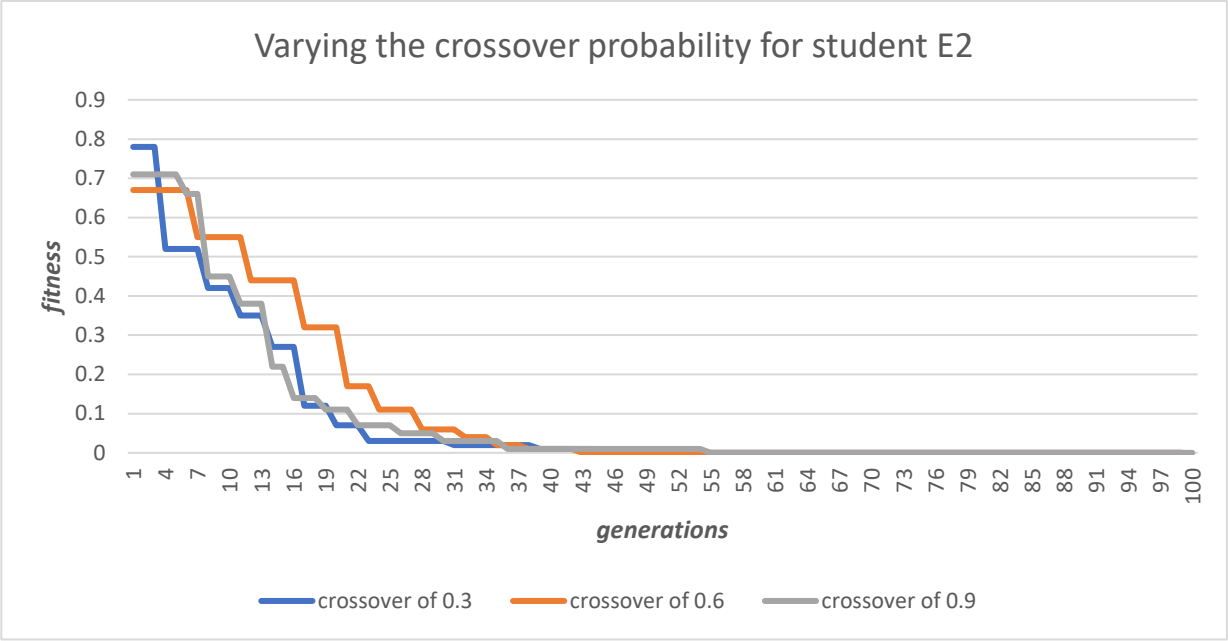

Fig. 7 Varying the crossover probability for student E2.

The second parameter to experiment was the mutation probability, which was tested as 0.03, 0.06, and 0.09; with a fixed crossover probability of 0.03, see Fig 9. As can be seen, in all cases, the proposed method can generate an adequate PUCT for student E2.

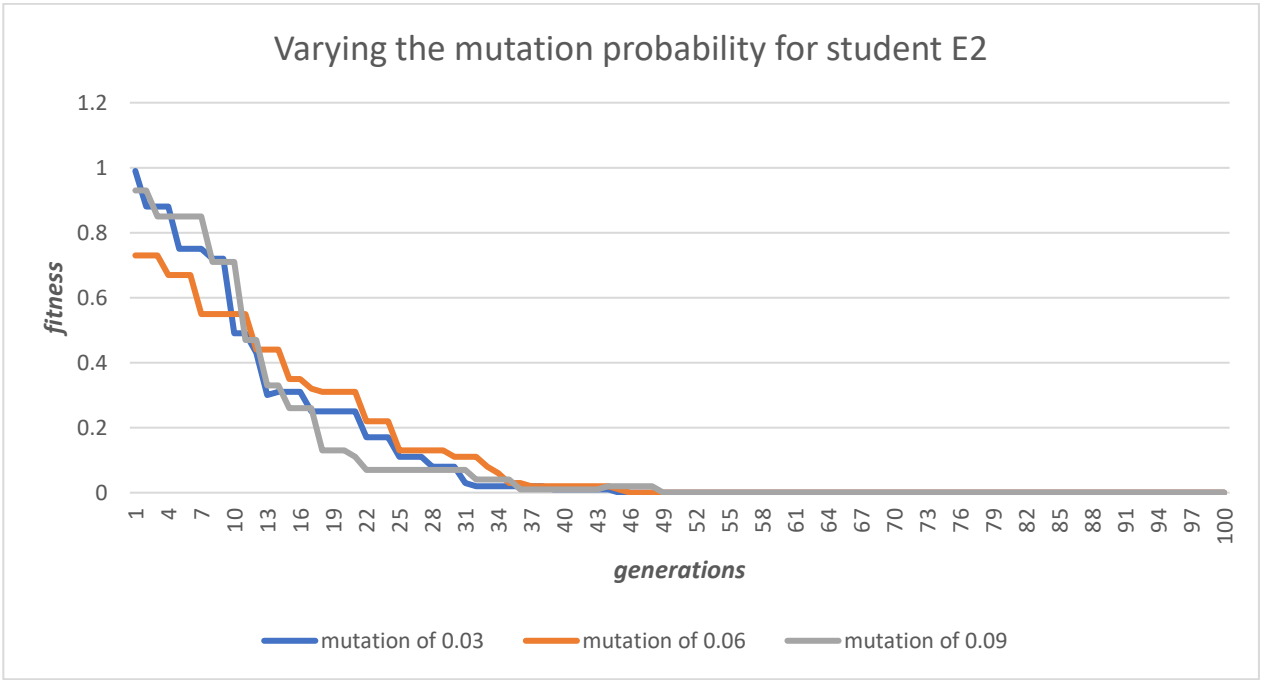

Fig. 8 Varying the mutation probability for student E2.

Fig. 10 shows the performance of the proposed GA using the crossover probability of 0.3 and the mutation probability 0.03. In order to observe if this behavior does not depend on the conditions of the student E2, the same scheme parameters were repeated on E3 student. Graphics 11 and 12 show the performance of the proposed GA using the crossover probability of 0.3 and the mutation probability of 0.03.

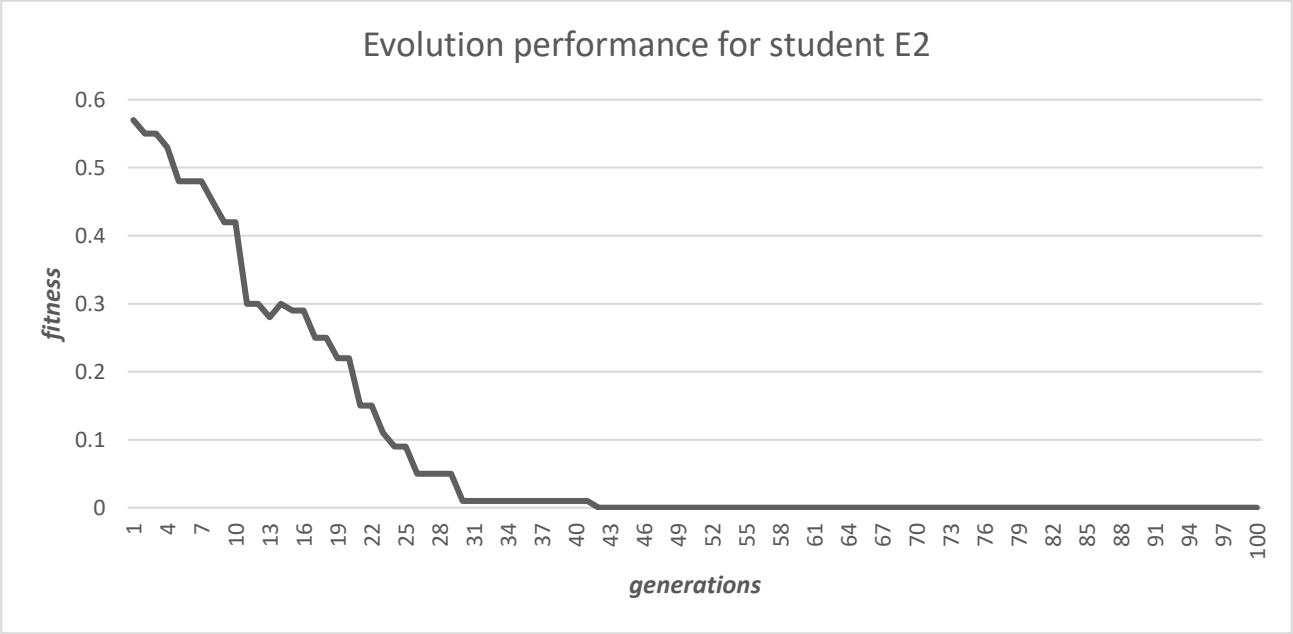

Fig. 10 Evolution performance for student E2.

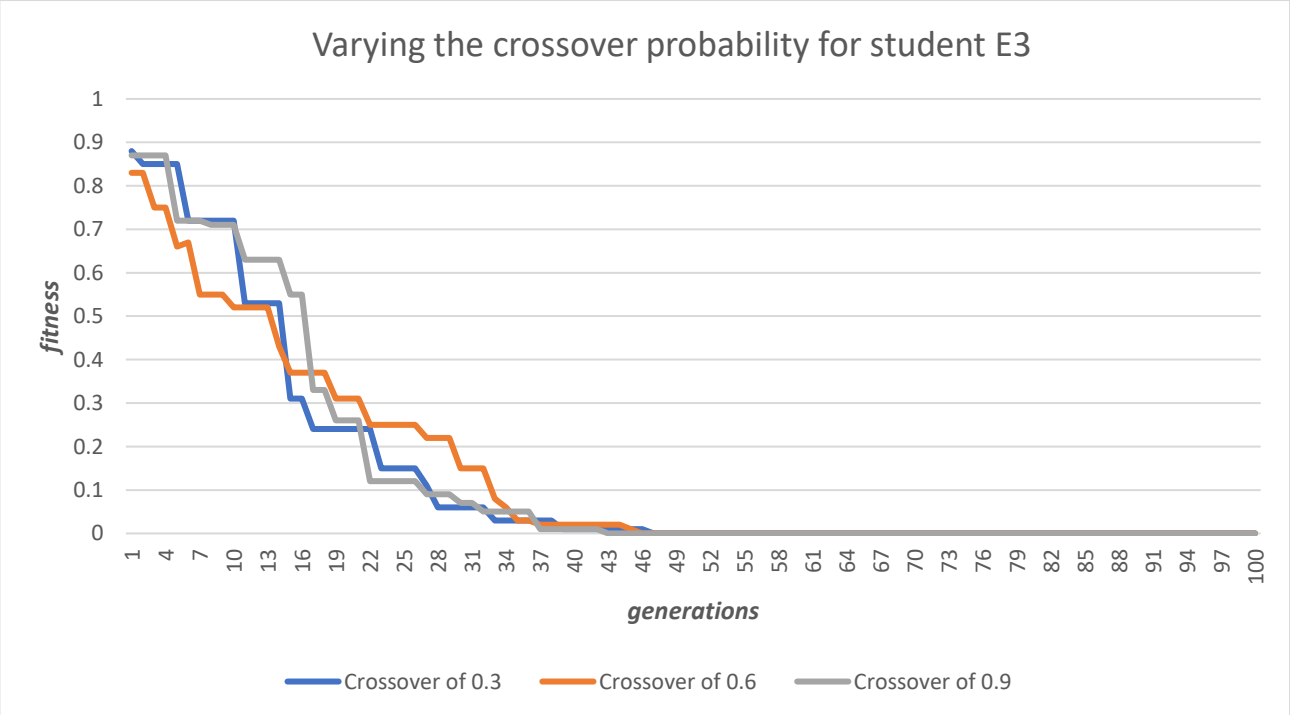

Fig. 9 Crossover probability for student E3.

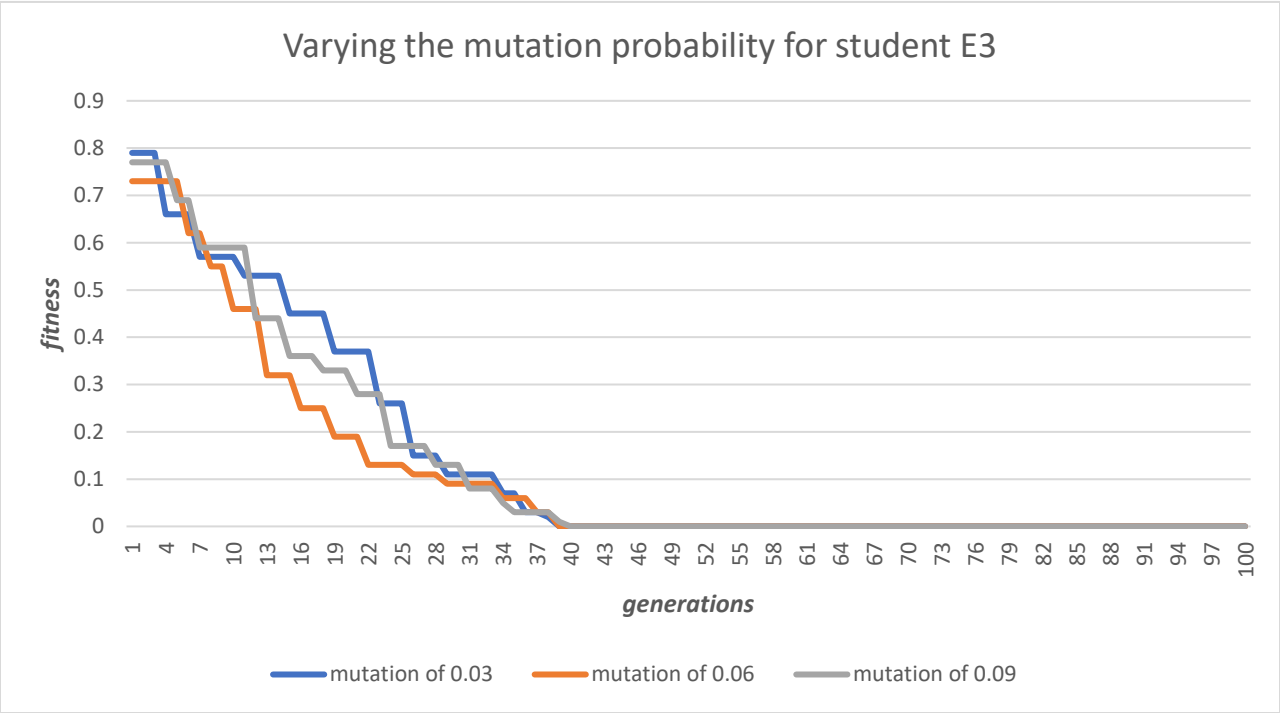

Fig. 10 Mutation probability for student E3.

Fig. 13 shows the performance of the proposed GA using the crossover probability of 0.3 and the mutation probability 0.03 for student E3. With the experimentation of this section, the crossover probability of 0.3 and the mutation probability 0.03 were used for all the experiments in the next section.

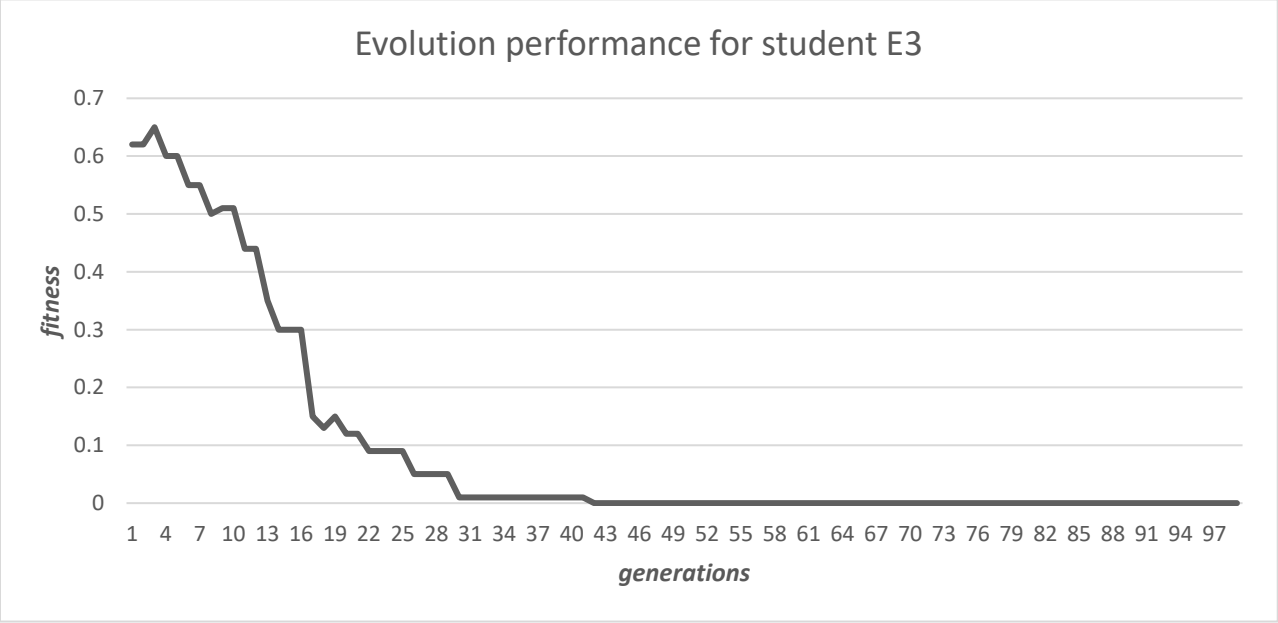

Fig. 11 Evolution performance for E3 student.

5.3 Results

Using the timetable constraints and preferences of students of Table 1, the genetic proposed method with the previously adjusted parameters obtains the fitness results shown in the middle column of automatic PUCT of Table 2. In Table 2, according to these results, our proposed method obtained equal or better results than the current manually-made PUCT. These results support our hypothesis about an evolutionary approach —as the well-known genetic algorithm is enough to find good PUCT based on the student preferences and curriculum constraints. In all the results, the variables α, β, γ, ε, λ, ψ, θ, and Ω of the fitness function receive the same weight (0.125).

Table. 2. Comparison of the manual and automatic PUCT with the fitness function for the 25 students of Table 1. Furthermore, the comparison shows, in the right column, the student point-of-view of the automatic PUCT with respect to their manual PUCT into four categories: worse, equal, equivalent and better

Student	Fitness Evaluation		Qualitative Evaluation
	Manual PUCT	Automatic PUCT	
E1	0	0	Equal
E2	0	0	Equal
E3	0	0	Equal
E4	0	0	Equivalent
E5	0	0	Equivalent
E6	0.07	0.05	Better
E7	0.08	0.06	Better
E8	0.02	0	Better
E9	0.08	0.04	Better
E10	0.02	0.01	Better
E11	0.08	0.03	Better
E12	0.06	0.02	Better
E13	0.06	0.04	Better
E14	0.08	0.06	Better
E15	0.07	0	Better
E16	0.8	0	Better
E17	0.02	0	Better
E18	0.08	0	Better
E19	0.03	0	Better
E20	0.06	0	Better
E21	0.08	0	Better
E22	0.07	0	Better
E23	0.08	0	Better
E24	0.08	0	Better
E25	0.07	0	Better

However, the quantitative values of Table 2 do not show the point-of-view assessment of the student about automatic PUCT. In the last column of Table 2, the student qualified the automatic PUCT with respect to the manually-made within four categories worse, equal, alternative and better. Although the qualitative evaluation is implicit in the quantitative evaluation, the qualitative evaluation shows that for students E4 and E5 (with the same fitness evaluation) the automatic PUCT was equivalent; and for students E6, E7, E8, E10, E13, E14, and E17 the difference between the fitness evaluation is too close (less than or equal to 0.02) the students note the difference since it could be labeled as "equivalent". Fig. 14 shows the qualitative evaluation approved by the students of the PUCT generated by the proposed method. In 12% of the results, the automatic PUCTs were the same as those made by the student. In 8% of the results, the automatic PUCTs were equivalent to those carried out by the student and are a feasible alternative. In 80% of the results, better PUCTs were created than those generated manually by the student; it should be noted that in no case were PUCTs worse than those generated manually or with overlapping timetables. These results approve our hypothesis about the fitness function is formulated in relation with the point of view of the students.

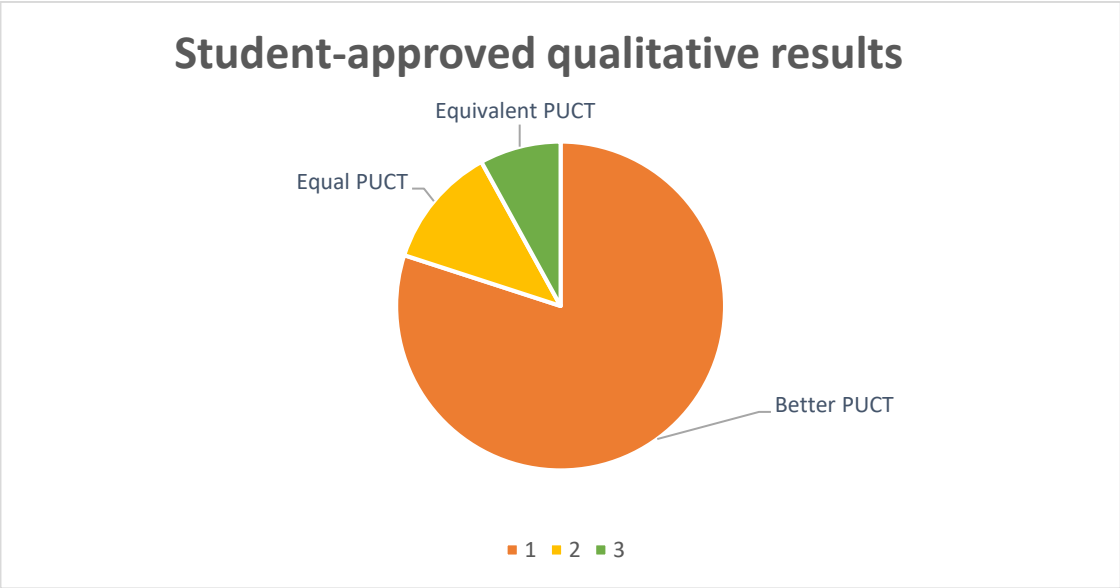

Fig. 12 Qualitative results.

Further experimentation with the Simulated-Annealing Algorithm (SAA) was performed to this problem. We selected the meta-SAA heuristic because it could be a feasible alternative for the implementation on the mobile devices of the students. In this case, the initial parameters for simulated annealing are: initial temperature of 2, minimum temperature of 1/10000, and maximum number of iterations of 100, with the geometric cooling function. In Fig 15, the performance of the SAA for the students E1, E2 and E3 is shown. In all three cases, the SAA performed well with the advantage of using only a single solution, therefore the SAA algorithm required fewer evaluations than the genetic algorithm.

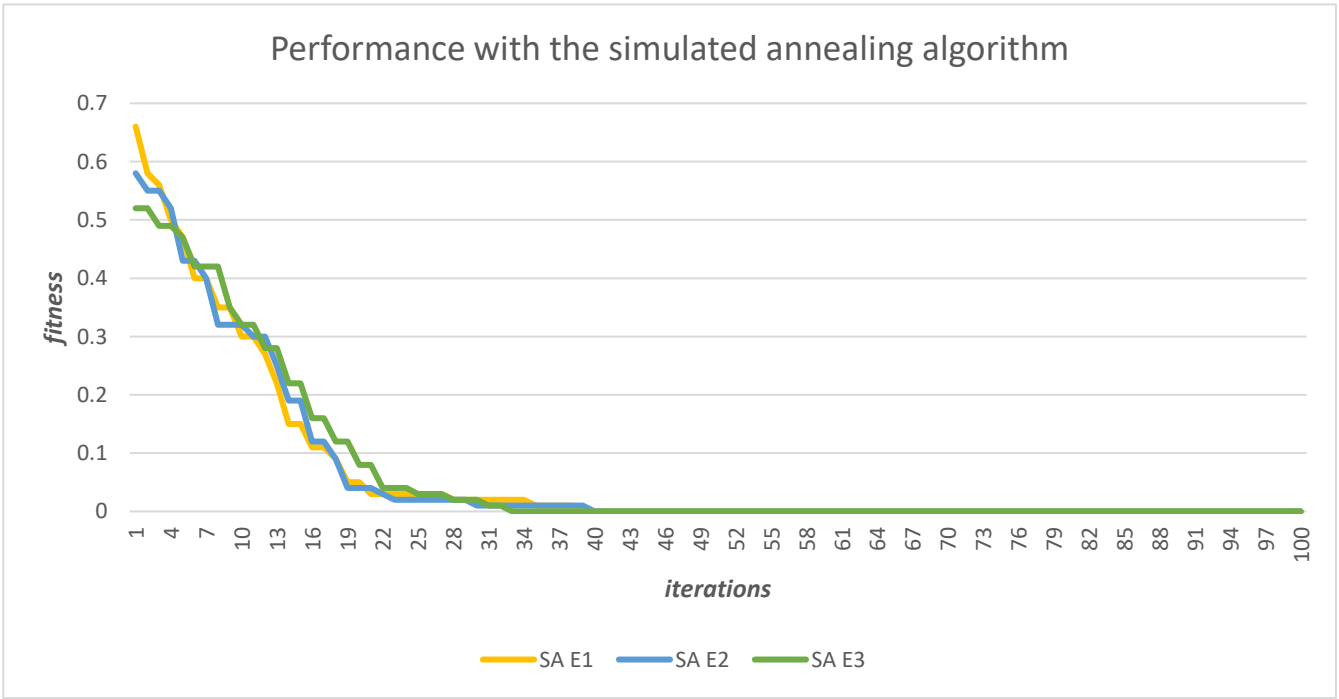

Fig. 15 Performance with simulated annealing algorithm for the students E1, E2 and E3.

6 Conclusions

It is worth mentioning that the educational performance of an institution depends on the educational performance of the student, which in turn depends on the construction of a good personal timetable, which is a time-consuming task. The first contribution of this work is to address a new educational timetabling problem that consists of generating a personal university course timetabling for students from a published course catalog. The second contribution is that we present the formulation of the fitness function that considers the constraints and preferences presented by UAEMEX students in relation with the student point-of-view. The third contribution relies on the use the well-known genetic-based method for generating a personal university course timetabling. With the formulation of our fitness function and the use of a well-known evolutionary algorithm, the proposed method can be easily adapted to other higher education institutions.

According to experimentation with the ground truth dataset, the proposed method shows good qualitative results based on the evaluation of the students. In all the cases, our method presents a high-quality PUCT that, compared to the students it generates the same, equivalent, or better PUCTs, but it was never worse. The creation of a test data set is a time-consuming task, so in the future, it is expected to automate this task to perform a greater number of tests. Further experimentation with the simulated annealing algorithm suggests that it could be a viable alternative to be implemented on mobile devices. In the future, it will be useful to make comparisons with other meta-heuristic algorithms.

References

1. A. Hertz, “Finding a feasible course schedule using Tabu search,” Discret. Appl. Math., vol. 35, no. 3, pp. 255–270, 1992, doi: 10.1016/0166-218X (92)90248-9.
2. A. Hertz, “Tabu search for large scale timetabling problems,” Eur. J. Oper. Res., vol. 54, no. 1, pp. 39–47, Accessed: Jan. 11, 2021. https://www.academia.edu/20101985/Tabu_search_for_large_scale_timetabling_problems.
3. A. T. Ubando, A. B. Culaba, K. B. Aviso, D. K. S. Ng, and R. R. Tan, “Fuzzy multi-objective approach for designing of biomass supply chain for polygeneration with triple footprint constraints,” in ASME International Mechanical Engineering Congress and Exposition, Proceedings (IMECE), Apr. 2013, vol. 12, doi: 10.1115/IMECE2013-66236.
4. C. Cotta and A. J. Fernández, “Memetic algorithms in planning, scheduling, and timetabling,” Stud. Comput. Intell., vol. 49, pp. 1–30, 2007, doi: 10.1007/978-3-540-48584-1_1.
5. D. Abramson and J. Abela, “A parallel genetic algorithm for solving the school timetabling problems,” Proc. First Int. Work. Parallel Process. AI, pp. 6–11, 1991.
6. D. Abramson, M. Krishnamoorthy, and H. Dang, “Simulated annealing cooling schedules for the school timetabling problem,” Asia-Pacific J. Oper. Res., vol. 16, no. 1, pp. 1–22, 1999.
7. E. Burke, “Examination timetabling in British universities: A survey,” Lect. Notes Comput. Sci., vol. 1153, pp. 76–90, 1996, doi: 10.1007/3-540-61794-9_52.
8. E. Jansen, “Practices in timetabling in higher education institutions: A systematic review,” PATAT 2016 - Proc. 11th Int. Conf. Pract. Theory Autom. Timetabling, pp. 295–316, 2016.
9. E. K. Burke and S. Petrovic, “Recent research directions in automated timetabling,” Eur. J. Oper. Res., vol. 140, no. 2, pp. 266–280, 2002, doi: 10.1016/S0377-2217(02)00069-3.
10. E. K. Burke, J. R. Newall, and R. E. Weare, “A memetic algorithm for university exam timetabling,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 1153, pp. 241–250, 1996, doi: 10.1007/3-540-61794-9_63.
11. F. Vericat, C. O. Stoico, C. M. Carlevaro, and D. G. Renzi, “Genetic algorithm for the distribution function of the electron gas,” Interdiscip. Sci. Comput. Life Sci., vol. 3, no. 4, pp. 283–289, 2011, doi: 10.1007/s12539-011-0108-3.
12. G. Cubillos, “Timetabling School problem and genetic algorithms:” Vínculos, vol. 10, no. 2, pp. 259–276, 2013.
13. Henderson, D., Jacobson, S. H., & Johnson, A. W. (2006). The Theory and Practice of Simulated Annealing. In Handbook of Metaheuristics (Issue April). https://doi.org/10.1007/0-306-48056-5_10
14. J. A. Ferland and A. Lavoie, “Exchanges procedures for timetabling problems,” Discret. Appl. Math., vol. 35, no. 3, pp. 237–253, 1992, doi: 10.1016/0166-218X (92)90247-8.
15. J. Gómez, “Application of Genetic Algorithms Technique in the Generation of Academic Schedules,” KnE Eng., vol. 2020, pp. 150–165, 2020, doi: 10.18502/keg.v5i1.5927.
16. Jansen, E. (2016). Practices in timetabling in higher education institutions: A systematic review. PATAT 2016 - Proceedings of the 11th International Conference on the Practice and Theory of Automated Timetabling, 295–316.
17. L. De Sardi, “Evolutionary Multi-objective Optimization by a model of nonlinear,” 2011.
18. M. W. Carter, “A comprehensive course timetabling and student scheduling system at the University of Waterloo,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 2079 LNCS, pp. 64–82, 2001, doi: 10.1007/3-540-44629-x_5.
19. P. Alvarado-moya, “Multiobjective optimization with expensive functions. Survey on the state of the art,” pp. 16–24, 2016.
20. P. B. Myszkowski and M. E. Skowroński, “Hybrid ant colony optimization in solving multi-skill resource-constrained project scheduling problem,” Soft Comput., vol. 19, no. 12, pp. 3599–3619, Dec. 2015, doi: 10.1007/s00500-014-1455-x.

21. P. Boizumault, “Constraint logic programming for examination timetabling,” J. Log. Program., vol. 26, no. 2 SPEC. ISS., pp. 217–233, 1996, doi: 10.1016/0743-1066(95)00100-x.
22. R. Weare, E. Burke, and D. Elliman, “A Hybrid Genetic Algorithm for Highly Constrained Timetabling Problems School of Computer Science and Information Technology University of Nottingham Computer Science Technical Report No. NOTTCS-TR-1995-8 A Hybrid Genetic Algorithm for Highly Constrain,” no. January 1995, 2017.
23. S. Daskalaki, T. Birbas, and E. Housos, “An integer programming formulation for a case study in university timetabling,” Eur. J. Oper. Res., vol. 153, no. 1, pp. 117–135, 2004, doi: 10.1016/S0377-2217(03)00103-6.
24. S. Petrovic and E. Burke, “University timetabling,” Handb. Sched. Algorithms, Model. Perform. Anal., no. December, pp. 45-1-45–24, 2004.
25. T. Frühwirth and S. Abdennadher, “University Course Timetabling,” pp. 117–122, 2003, doi: 10.1007/978-3-662-05138-2_17.
26. X. Chen, Y.-S. Ong, M.-H. Lim, and K. C. Tan, “A multi-Facet survey on memetic computation,” IEEE Trans. Evol. Comput., vol. 15, no. 5, p. 591, 2011, doi: 10.1109/TEVC.2011.2132725.
27. Z. Lü and J. K. Hao, “Adaptive Tabu Search for course timetabling,” Eur. J. Oper. Res., vol. 200, no. 1, pp. 235–244, 2010, doi: 10.1016/j.ejor.2008.12.007.