www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 17(1), Jan-April 2026, 263-274. ISSN: 2007-1558. https://doi.org/10.61467/2007.1558.2026.v17i1.1144

Repair and Initialization Functions in the Generation of School Schedules with Genetic Algorithms

Alejandro Moreno Martinez[1], *Victor Manuel Landassuri Moreno*[1], *Asdrúbal López Chau*[2], *Saturnino Job Morales Escobar*[1]
[1] Centro Universitario UAEM Valle de México, Boulevard Universitario S/N, Río San Javier, 54500, Cd. López Mateos, Atizapán de Zaragoza, Méx.
[2] Centro Universitario UAEM Valle de Zumpango, Zumpango de Ocampo, 55600, Méx.
amorenom758@alumno.uaemex.mx, {vmlandassurim, alchau, sjmoralese}.@uaemex.mx

Abstract. The elaboration of school timetables is a complex and laborious task when performed manually, due to the large number of requirements and constraints that must be considered for its correct construction. In this work, the problem is addressed by using a simple Genetic Algorithms (GA) and compared with an improved approach that incorporates both an initialization function and a repair function. The study uses real data from the Computer Engineering course at the Centro Universitario UAEM Valle de México. The results obtained show that the initialization function significantly reduces errors from the first generations, while the repair function further accelerates the reduction of class or teacher splices. Thus, the effectiveness of the proposed approach to solve the scheduling problem in the university center is demonstrated.

Keywords: Genetic Algorithms; School Schedules; Timetabling

Article Info
Received June 2, 2025
Accepted July 10, 2025

1 Introduction

The problem of generating class schedules is a common challenge in academic planning, involving variables such as classrooms, subjects, teachers, schedules (day and time), and student groups. These variables must be adjusted to avoid conflicts or inconsistencies—such as two groups being assigned to the same room at the same time—thus making it a search and optimization problem.

Timetable generation is classified as an NP-complete problem (non-deterministic polynomial time) (Guerra et al., 2013); thus, GAs are considered an ideal tool for solving it. It is worth noting that if a solution (i.e., a pre-existing timetable) already exists and the problem remains unchanged, there is no need to search for a new one. However, if any changes occur, adjustments are necessary to obtain an acceptable solution. These changes usually arise due to modifications in teaching staff or other factors beyond the control of individuals or institutions.

To avoid such situations, both hard and soft constraints must be satisfied. Examples of hard constraints include: 1) classroom size, where small groups should not be assigned to large classrooms and vice versa; 2) teacher availability, considering that teachers may have responsibilities at other institutions; and 3) the teacher's area of expertise, since, for example, a mathematics teacher should not be assigned to teach a social sciences class.

On the other hand, soft constraints may include: a) scheduling certain classes in the early hours of the day, considering the cognitive benefits for students who are more alert in the morning compared to later in the day; b) teacher preferences for teaching specific subjects; c) schedule continuity, avoiding gaps (i.e., idle or "dead" hours) between classes; and d) the total number of subjects a student takes in a single day.

It is important to note that soft constraints may or may not be satisfied, and yet the solution can still be considered valid — that is, one that can be implemented in a real-world context and functions adequately across the various aspects considered.

There are various techniques to address the problem of class schedule generation. Some of the approaches that have been used include Genetic Algorithms (GAs) (Guerra et al., 2013), local search algorithms (Shahvali Kohshori & Saniee Abadeh, 2012), linear programming through mathematical models, and heuristic methods (Abdullah et al., 2012).

2 Generation of School Schedules with Genetic Algorithms

The use of Genetic Algorithms (GAs) for schedule design has been explored in various ways and applied across multiple fields. For example, in the medical field, GAs have been used to plan surgery staff schedules (Bejarano, 2011) and to improve the efficiency of nursing staff schedules (Jafari & Salmasi, 2015). They have also been applied to design train schedules, or to program train routes by adding extra trains to a cyclic schedule, identifying the IDs of both original and additional trains at the station, and treating those IDs as genes (Tan et al., 2020), among other applications.

As mentioned before, GAs provide an efficient way to solve problems. In this sense, (Castrillón, 2014) utilized a GA in conjunction with a random operator to avoid getting trapped in local optima. When the algorithm becomes stuck in one of them, a completely random individual is generated, allowing the exploration of new routes. On the other hand, Abramson and (Abela, 1991) pointed out that GAs face the drawback of long execution times when run sequentially. To improve performance, they parallelized the algorithm by separating tasks that can be executed independently. As a result, the problem is solved faster than with a conventional GA.

In the work by (Jha, 2014), a GA was implemented for the generation of exam schedules. An initial population of feasible solutions was created randomly, taking into account constraints such as room availability, avoiding double bookings in classrooms, limiting exams to one per time slot, and ensuring that each subject has an assigned exam.

On the other hand, works such as that of (Cortez Vásquez et al, 2010), based on Genetic Algorithms, implement an approach that handles integer arrays in three phases: the first assigns teachers to classes, the second assigns schedules to classes, and the third assigns classrooms to classes. To avoid generating invalid individuals, the algorithm proposes the selective use of available resources — i.e., considering only the values that help satisfy specific constraints. For example, if there are 10 teachers but only 3 are qualified to teach calculus, the assignment is limited to those 3 instead of randomly selecting from all 10. This strategy generates an initial population that already satisfies certain constraints, thus saving time by directly excluding invalid solutions without the need for evaluation.

Likewise, (Abdelhalim and El Khayat, 2016) proposed a GA-based approach that implements a utilization crossover operator, which relies on utilization rates to measure the efficiency of space usage. The goal is to avoid scheduling events with occupancy rates below 75% or above 100%, and instead, prioritize events that optimize the utilization of available space. This is achieved through a custom adaptivity function. When selecting an event, its utilization rate is calculated, and a move is executed only if the new location improves this rate without violating any hard constraints. If the move is successful, the event is removed from its original position when generating the new chromosome, thereby avoiding duplication. Similarly, targeted mutation modifies specific parts of the chromosome, performing the process in a way similar to local search. Like the crossover operator, if the mutation results in successful moves, the changes are applied to the chromosome; otherwise, no modification is made.

In the paper by (Mahlous and Mahlous, 2023), student preferences are treated as soft constraints for schedule generation. The authors employ initialization and repair functions to accelerate the convergence of the algorithm. Additionally, the computation of both the adaptivity and repair functions is parallelized, which contributes to improved algorithm performance. The study reports that more than 90% of student preferences are successfully satisfied. Regarding the selection mechanism, various methods such as roulette and rank selection are mentioned; however, the efficiency of tournament selection is particularly emphasized.

Meanwhile, (Zandavi et al., 2021) propose a hybrid optimization algorithm called the Simplex Non-dominated Sorting Genetic Algorithm (SNSGA) for solving the multi-user remote laboratory scheduling problem. This algorithm combines the Nelder–Mead Simplex algorithm with the Non-dominated Sorting Genetic Algorithm (NSGA), enhancing both exploration and exploitation capabilities. Experimental results indicate that SNSGA outperforms other heuristic algorithms in terms of efficiency.

Similarly, (Hafsa et al., 2023) address the scheduling problem at Mandarine Academy using a multi-objective evolutionary approach. The NSGA-II genetic algorithm was implemented with the goal of generating optimal or near-optimal solutions. Additionally, a comparison is made between different evolutionary algorithms applied to the scheduling problem. The results indicate that NSGA-III outperforms the others in terms of convergence.

However, it is important to note that the efficiency of such systems depends heavily on the data, as discussed by (Savdekar et al., 2023), who developed a GA aimed at automating the schedule creation process, thereby reducing both the time and human effort required. The system provides flexibility and simplifies the scheduling process, although it demands considerable computational resources. Thus, Table 1 presents a comparison of their core methodologies, distinguishing characteristics, and the ways in which they differ from the present study.

Table 1 Related works in the domain of scheduling with genetic algorithms.

Author and Year	Main Technique	Key Characteristics	Difference from This Work
Cortez Vásquez et al. (2010)	GA in 3 sequential phases	Sequential assignment of teachers, schedules, and classrooms	No use of guided initialization or repair functions
Abdelhalim & El Khayat (2016)	GA with utilization-based operator	Space optimization using adaptive crossover based on occupancy rate	Does not handle teacher availability conflicts directly
Mahlous & Mahlous (2023)	GA with initialization and repair	Student preferences considered; parallel computation	Preferences not considered here; focus on guided initialization/repair
Zandavi et al. (2021)	Hybrid GA (SNSGA)	Combines NSGA with Simplex for remote lab scheduling	Different domain; does not implement repair functions
Hafsa et al. (2023)	NSGA-II / NSGA-III	Comparison of multi-objective evolutionary algorithms	This work uses S-GA and GA-GIRF; not multi-objective

3 Genetic Algorithm Design

For schedule generation, a Simple Genetic Algorithm (S-GA) was implemented — i.e., it consists solely of the basic genetic operators and employs random initialization of individuals. This serves as a baseline for performance comparison with the proposed approach, which maintains the same structure as the S-GA but incorporates a guided initialization function to reduce as many errors as possible.

Additionally, an individual repair function was developed and implemented to accelerate the process and enable the algorithm to solve the problem in fewer generations. This enhanced version will be referred to as the Genetic Algorithm with Guided Initialization and Repair Functions (GA-GIRF). **Fig. 1** shows the general diagram.

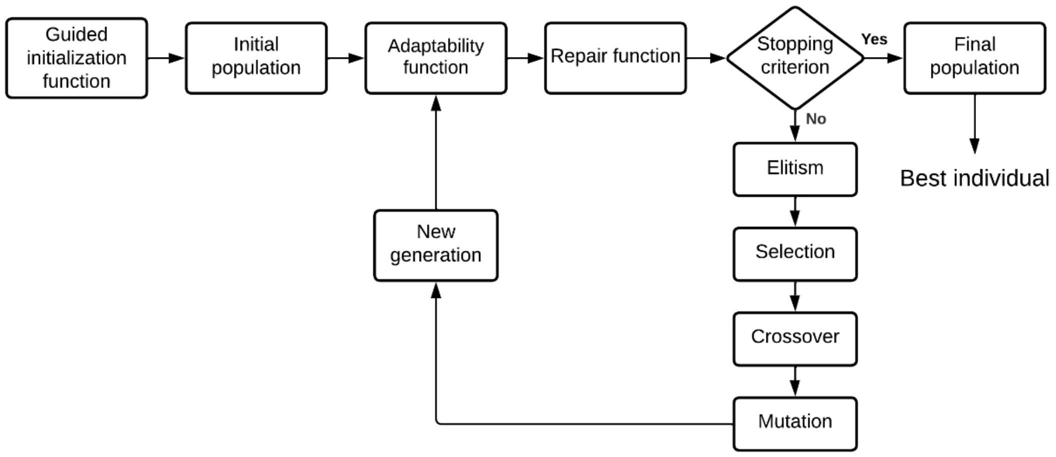


Fig. 1 GA-GIRF Flowchart

Initially, the implementation of the repair function was considered after the genetic operators of selection, crossover, and mutation, in order to correct the individuals. However, this approach could introduce additional delays in the algorithm, as it would require evaluating the population three times.

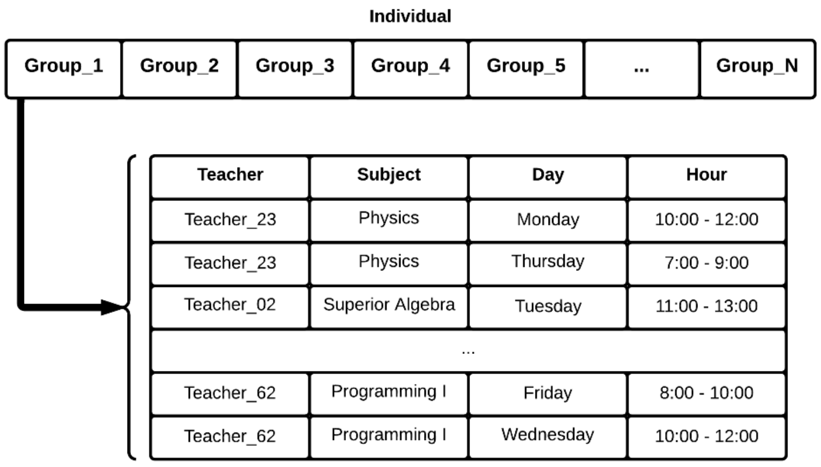
3.1 Constraint Definition

A set of constraints that the genetic algorithm must satisfy to produce feasible solutions is considered. These constraints are as follows:

- There must be no overlapping schedules for any resource.
- Each resource includes the teacher, class hours, and subject.
- Each faculty member has a maximum number of class hours they are allowed to teach.
- A teacher cannot be assigned to just any subject; subject-specific expertise must be respected.
- Each subject must fulfill the required number of hours per week.
- Teachers assigned to a course must be available during the scheduled time slots.

3.2 Chromosome Representation

To solve the scheduling problem using a GA, it is necessary to express the problem in terms of chromosomes. In the present work, each group is represented as a chromosome, as illustrated in **Fig. 2**. Therefore, if there are *N* groups, there will be *N* chromosomes in each individual. In this representation, each group consists of a list of data elements.



Teacher	Subject	Day	Hour
Teacher_23	Physics	Monday	10:00 - 12:00
Teacher_23	Physics	Thursday	7:00 - 9:00
Teacher_02	Superior Algebra	Tuesday	11:00 - 13:00
...			
Teacher_62	Programming I	Friday	8:00 - 10:00
Teacher_62	Programming I	Wednesday	10:00 - 12:00

Fig. 2. Individual Structure

3.3 Initialization function

In GAs, the first generation of individuals can be created randomly or, alternatively, in a guided manner through a specific algorithm that generates individuals under certain conditions. The goal is to accelerate the process so that the GA can solve the problem in fewer generations. Several studies describe different approaches to generating the initial population. For example, in (Flores et al., 2004), a binary string representation is used in which all values are initially set to zero and then modified. In contrast, (Assi et al., 2018) initializes individuals with a single one per column.

Other works, such as (Shahvali Kohshori & Saniee Abadeh, 2012), use matrices to generate schedules, verifying both the availability and experience of teachers, as well as the availability of classrooms before assignment.

In (Mahlous & Mahlous, 2023), a function is proposed that allows the generation of feasible initial solutions, helping the algorithm focus on satisfying soft constraints.

To create a schedule, one subject unit from the desired semester is selected from a list of available subjects. Then, from a list containing information about which subjects each teacher is qualified to teach, a teacher is selected. The algorithm then checks whether the selected teacher is available to teach during a specific shift (morning or afternoon). If availability is confirmed, a time slot within that period is randomly chosen. The day of the week is also randomly selected. If this is the first assignment, it is directly placed into the schedule. Otherwise, if there is a time conflict, the algorithm attempts to assign the class on a different day, in order to generate a solution that is as close as possible to the ideal, as illustrated in **Fig. 3**.

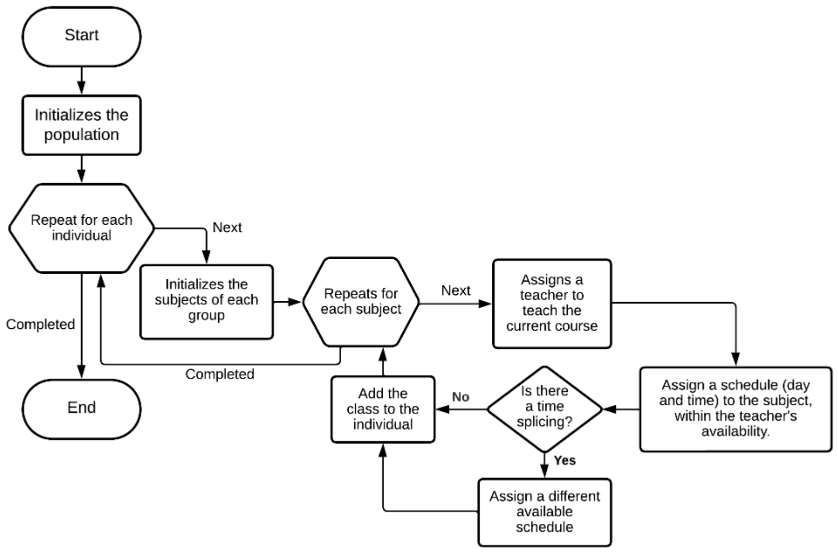

Start
Initializes the population
Repeat for each individual
Next
Completed
End
Initializes the subjects of each group
Repeats for each subject
Next
Completed
Assigns a teacher to teach the current course
Assign a schedule (day and time) to the subject, within the teacher's availability.
Is there a time splicing?
No
Yes
Add the class to the individual
Assign a different available schedule

Fig. 3. Initialization Function

3.4 Adaptability function

The global adaptability function *F_g(i)* evaluates each individual *i* within the population. Equation (1) calculates this value as the summation of the local functions (f_l) of the groups that make up the individual.

$$F_g(i) = \sum_{x=1}^{N} f_l(x) + EDG(i) + ESH(i) \tag{1}$$

Where:

- i: Represents the individual within the population.
- *x*: Corresponds to the groups that conform to the individual (*i*). Everyone is composed of (*N*) groups, and (f_l) evaluates each of these groups.
- N: Is the total number of groups belonging to the individual *i*.
- EDG(i): Penalizes splices between teachers assigned in two or more groups at the same time.
- ESH(i): Penalizes overtime, which occurs if more hours are assigned than the teacher is entitled to.

The function $F_g(i)$ evaluates the quality of the individual's overall schedule by integrating both local and global constraints. Equation (1) is adapted from (Broca, 2016), with modifications to include splice penalties (EDG) and teaching overload (ESH), reflecting the specific constraints of this study.

For the local adaptability function $f_l(x)$ is applied to each group that makes up individual *i*. A series of parameters are evaluated using Equation (2) to determine whether each group is valid or not. This formulation extends the local evaluation function proposed by (Broca, 2016), incorporating additional terms such as teacher incompatibility (DIM) and group-level hour validations (HSG, HAM).

$$f_l(x) = EC(x) + DID(x) + EHD(x) + DIM(x) + HSG(x) + HAM(x). \tag{2}$$

Where:

- EC: Represents class splicing where two or more classes are scheduled at the same time.
- DID: Represents whether the teacher has the availability to teach a subject at the assigned time.
- EHD: Indicates when two or more teachers are assigned to the same group at the same time.
- DIM: Represents that a teacher cannot teach the assigned subject, this error applies to evaluate the simple GA, due to the use of a guided initialization function manages to avoid being generated.
- HSG: Checks that the group meets the required weekly hours, e.g., 32 hrs per week between all subjects.
- HAM: Verifies that each subject scheduled in the group has the hours specified in the Program of Study, e.g., that each subject meets 4 hrs per week.

3.5 Crossover operator

For this operator, uniform crossover was used. Two individuals are randomly selected from the list of parents. Each chromosome (or group) in these individuals has a 70% probability of being exchanged. This results in the generation of two new individuals, which are added to a new list representing the offspring. This process is repeated until a list of the same size as the original population is obtained, as illustrated in **Fig. 4**.

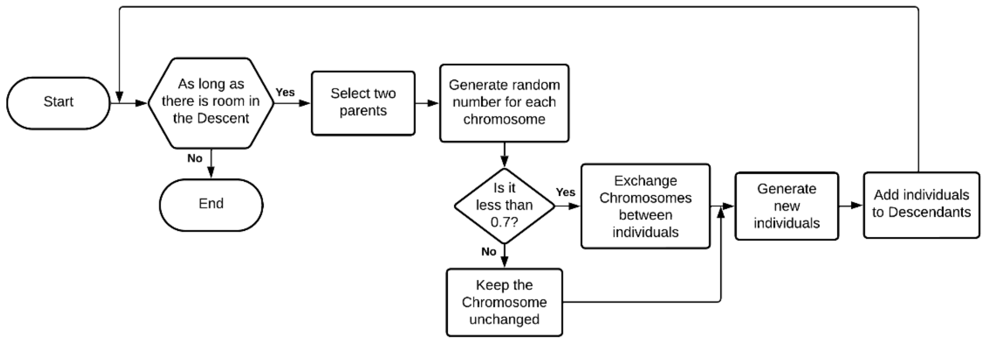

Start
As long as there is room in the Descent
Yes
No
End
Select two parents
Generate random number for each chromosome
Is it less than 0.7?
Yes
No
Exchange Chromosomes between individuals
Keep the Chromosome unchanged
Generate new individuals
Add individuals to Descendants

Fig. 4. Uniform Crossover Application Diagram

3.6 Mutation operator

The mutation operator is set with a 30% probability of altering the value of one or more sub-genes, such as teacher, day, or time. All individuals in the offspring may undergo one or more mutations, except for those selected by elitism. The mutation process iterates through all the genes (i.e., schedules) of each individual. Using random numbers, it determines whether each sub-gene will be modified. If a sub-gene is selected for mutation, its list of possible values is randomly shuffled to avoid consistently selecting the same value. This strategy enhances the exploration of a wider range of possibilities. As a result, each gene may undergo changes to its teacher, day, and time values, or remain unchanged.

In this work, and for the purpose of evaluating the GA's behavior, this operator is referred to as Random Mutation (*RM*). It is compared with a second variant called Restrictive Mutation (*ReM*), where instead of selecting from the entire pool of teachers, only those who are qualified to teach the corresponding subject are considered during mutation.

3.7 Repair functions

The use of genetic operators can cause individuals that were previously fit to become unfit after crossover or mutation. To address this issue, the implementation of repair algorithms to correct invalid offspring has been discussed in the literature, as a way to avoid discarding potentially valuable future solutions. For example, in (Mahlous & Mahlous, 2023), beyond the use of an initialization function, a repair function is proposed to correct time clashes, assign missing classes, and reassign students to appropriate schedules.

In this work, each individual's timetable is reviewed and evaluated using an error matrix. During this process, modifications are made to the schedule to correct specific types of errors. For example, in the case of class splicing errors (*EC*), changes are made to the time or day of the class. For teacher availability errors (*DID*), a new time slot within the teacher's available hours is assigned. In cases of teacher overlap between different groups, a replacement teacher who is qualified to teach the subject is searched for and assigned. If the modification results in an improvement (i.e., it reduces the total error according to the error matrix), the change is implemented in the individual's schedule. If the modification is not beneficial (i.e., it does not reduce the error or worsens it), the original schedule is retained without changes. As shown in **Fig. 5**.

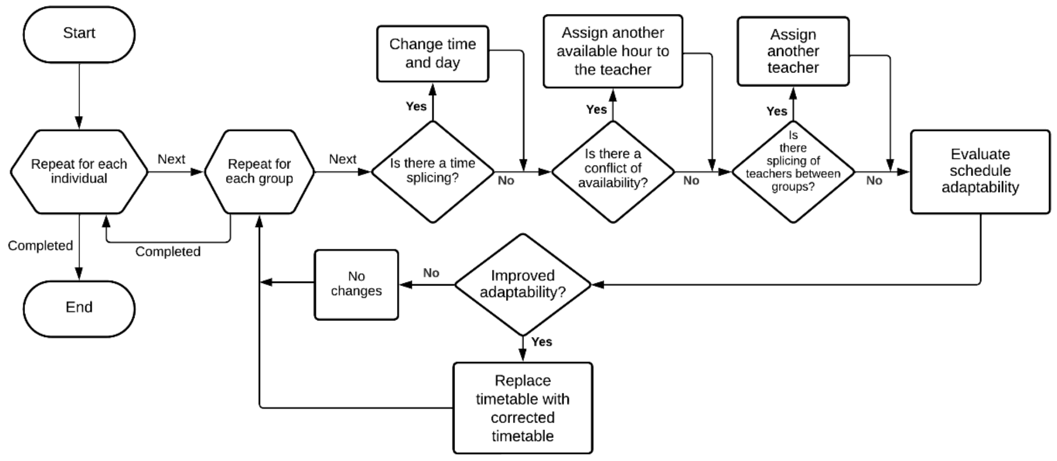

Start
Repeat for each individual
Next
Repeat for each group
Next
Is there a time splicing?
Yes
Change time and day
No
Is there a conflict of availability?
Yes
Assign another available hour to the teacher
No
Is there splicing of teachers between groups?
Yes
Assign another teacher
No
Evaluate schedule adaptability
Improved adaptability?
No
No changes
Yes
Replace timetable with corrected timetable
Completed
Completed
End

Fig. 5. Repair Function Flowchart

4 Experimental setup

To evaluate the performance of the GA, data from the Centro Universitario UAEM Valle de México (CU UAEM VM) were used, focusing specifically on the subjects offered in the Computer Engineering degree program (ICO).

Starting with the S-GA, we used a population of 100 individuals; a ranking selection that will take 75% of the population; a uniform cross with a probability of 70%; and mutations with a probability of 30%. In addition, elitism was applied, retaining the 10 best individuals. This process was repeated for 100,000 generations.

For the GA-GIRG, the parameters remained the same, with the only difference of reducing the number of generations to 10,000, after performing the first tests and observing that there was no significant error reduction.

All experiments were performed on a desktop computer equipped with an Intel(R) Core (TM) i7-4770 CPU and 20 GB of DDR3 RAM at 1333 MHz. The algorithms were implemented in the Spyder IDE (version 6.0.7) running on Python 3.11.12, with Pandas version 2.2.3 and Numpy version 2.2.6.

4.1 CU UAEM VM Dataset

For ICO, the ICO F19 curriculum map is considered, which consists of 10 periods where the student must study and accredit 60 learning units of the total of 65 learning units contained in the academic plan. Each period consists of a specific number of subjects and a total number of hours per week that must be completed.

The data set was manually constructed by collecting and organizing the actual schedules of the morning and afternoon groups. Specifically, from the schedules for the period 2022-B (August-December) for odd semesters (1, 3, 5, 7 and 9), and 2023-A (February-June) for even semesters (2, 4, 6, 8 and 10). This includes several cohorts of students enrolled in different semesters. For the purposes of this research, the analysis focuses exclusively on period B, which includes 32 unique subjects.
The dataset captures the following information:

- Subject information: Course name, assigned period, and total weekly hours per subject.
- Instructor assignments: Each subject is linked to a teacher qualified to teach it, based on their area of specialization.
- Teacher availability: Each teacher has defined time blocks (07:00–14:00 and/or 14:00–21:00) during which they are available to teach.
- Teaching load constraints: All teachers are limited to a maximum of 16 hours per week of instruction.
- Schedule structure: Class data includes day of the week, time slot, assigned group, and subject taught.

5 Results

5.1 Comparison Between Random and Restrictive Mutation

Initially, a comparison was performed between two types of mutation: Random Mutation (*RM*) and Restrictive Mutation (*ReM*), using the AG-S as the baseline for testing. The objective of these experiments was to evaluate the performance of both mutation operators. To this purpose, 10 independent runs were conducted, and the error of the best individual was averaged across generations. The results are presented in **Fig. 6**.

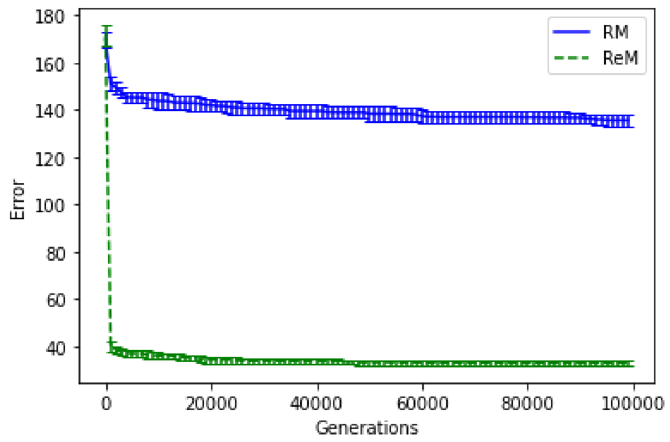


Fig. 6. Average Performance of the Best Individual Using *RM* and *ReM* (ICO, CU-UAEM-VM) with the AG-S

The ICO-CU-UAEM-VM dataset includes a large number of subjects and professors, many of whom are specialized and can only teach specific subjects. As a result, random modifications often generate invalid individuals, which slows down convergence. In contrast, Restrictive Mutation proves effective by correcting assignment errors related to teacher eligibility. Table 2 presents the average, standard deviation, minimum and maximum error of the best individual obtained over 10 independent runs, as well as the average execution time for the AG-S.

Table 2. Comparison of Best Results for the ICO-CU-UAEM-VM Dataset

	Mean	STD	Min	Max	Time(s)
Random Mutation (RM)	135.6	2.62	132	140	124,380
Restrictive Mutation (ReM)	33.0	0.89	31	34	115,354

According to the experiments conducted with RM and ReM, it can be concluded that mutation with restrictions helps reduce errors more effectively, allowing individuals to evolve better solutions in fewer generations. Additionally, it is faster compared to Random Mutation. Restrictive Mutation also enables the quick resolution of teacher scheduling conflicts within each group. The most common errors in the generated schedules are class time overlaps between subjects (*EC*) and teacher availability conflicts (*DID*).

Another important point to mention is that, due to the structure used for creating individuals, *HSG* and *HAM* errors do not occur during execution. This is because the crossover operator performs a complete exchange of groups (chromosomes) without altering the records (genes) that compose each group's schedule. Only the mutation operator modifies the sub-genes related to teacher, time, and day. However, the evaluation of *HSG* and *HAM* errors remains active during testing, which is beneficial in case the algorithm is extended in the future to support new configurations, constraints, or types of schedules. These errors may also appear in the event of incorrect initialization.

5.2 Simple Genetic Algorithm Results

Considering the results obtained from testing both mutation strategies, the following experiments were conducted using Restrictive Mutation. As before, 10 independent runs were performed. For each run, the best individual, the worst individual, and the average fitness per generation were recorded over 100,000 generations. At the end of the 10 runs, the data were averaged, and the results are presented in **Fig. 7**.

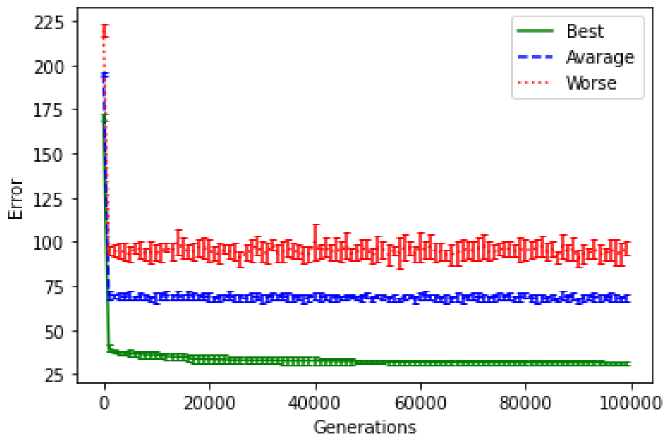
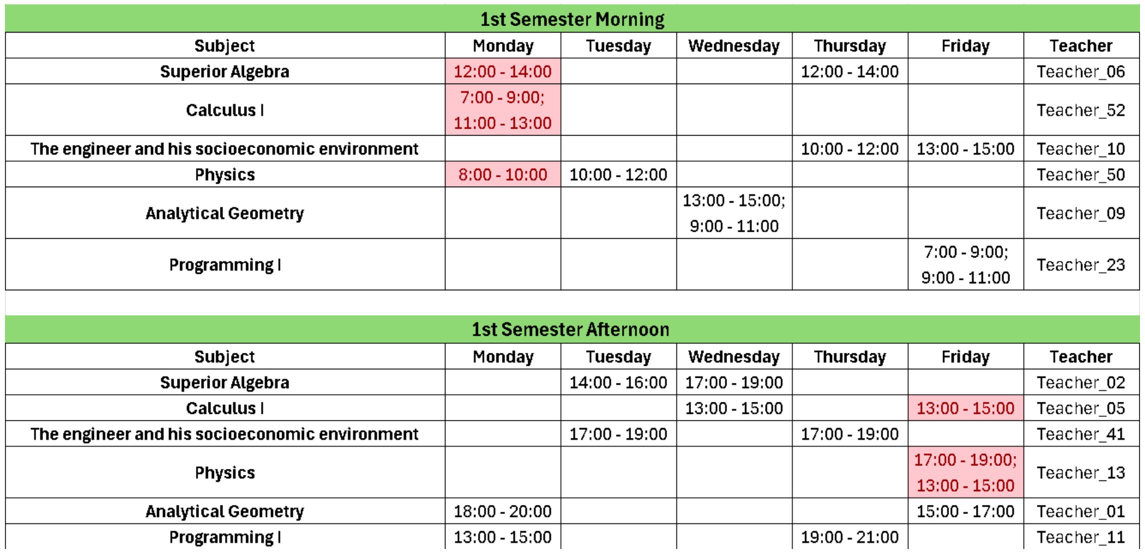


Fig. 7. Average Performance of the S-GA on the ICO-CU-UAEM-VM Dataset.

Fig. 7 shows a rapid decrease in error during the initial generations, followed by a stabilization in both the best and average individual performance. The worst individual, however, exhibits oscillations due to the effects of the crossover and mutation operators. It is important to note that not all combinations lead to improvements—some may introduce new errors that were not present before.

Additionally, **Fig. 7** displays vertical lines over each data point, which represents the standard deviation calculated from the 10 independent runs.

Using S-GA, it was observed that the algorithm was not able to find a complete solution. However, these tests are useful for analyzing the behavior of individuals across generations. Taking the best individual from one of the 10 independent runs, **Fig. 8** shows an example of a schedule generated by S-GA. In this example, three splicing errors can be seen in the Morning group (cells highlighted in red) and two in the Evening group.



1st Semester Morning

Subject	Monday	Tuesday	Wednesday	Thursday	Friday	Teacher
Superior Algebra	12:00 - 14:00			12:00 - 14:00		Teacher_06
Calculus I	7:00 - 9:00; 11:00 - 13:00					Teacher_52
The engineer and his socioeconomic environment				10:00 - 12:00	13:00 - 15:00	Teacher_10
Physics	8:00 - 10:00	10:00 - 12:00				Teacher_50
Analytical Geometry			13:00 - 15:00; 9:00 - 11:00			Teacher_09
Programming I					7:00 - 9:00; 9:00 - 11:00	Teacher_23

1st Semester Afternoon

Subject	Monday	Tuesday	Wednesday	Thursday	Friday	Teacher
Superior Algebra		14:00 - 16:00	17:00 - 19:00			Teacher_02
Calculus I			13:00 - 15:00		13:00 - 15:00	Teacher_05
The engineer and his socioeconomic environment		17:00 - 19:00		17:00 - 19:00		Teacher_41
Physics					17:00 - 19:00; 13:00 - 15:00	Teacher_13
Analytical Geometry	18:00 - 20:00				15:00 - 17:00	Teacher_01
Programming I	13:00 - 15:00			19:00 - 21:00		Teacher_11

Fig. 8. Example of a Schedule Generated by S-GA

5.3 GA-GIRF Results

This section presents the results obtained using the Genetic Algorithm with Guided Initialization and Repair Function (GA-GIRF). As in previous experiments, 10 independent runs were performed to average the algorithm's behavior. The resulting performance is shown in **Fig. 9**.

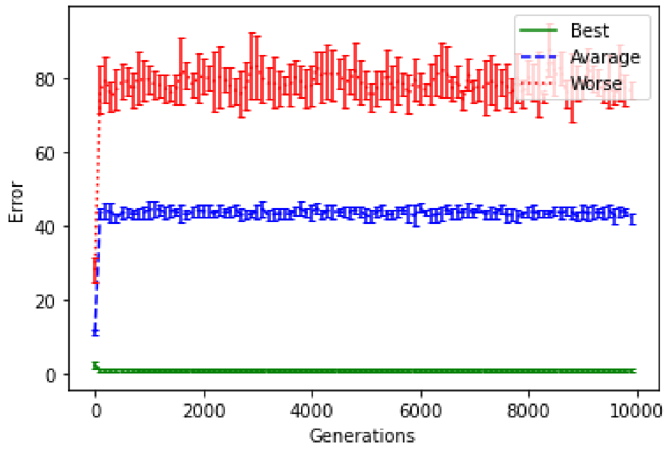
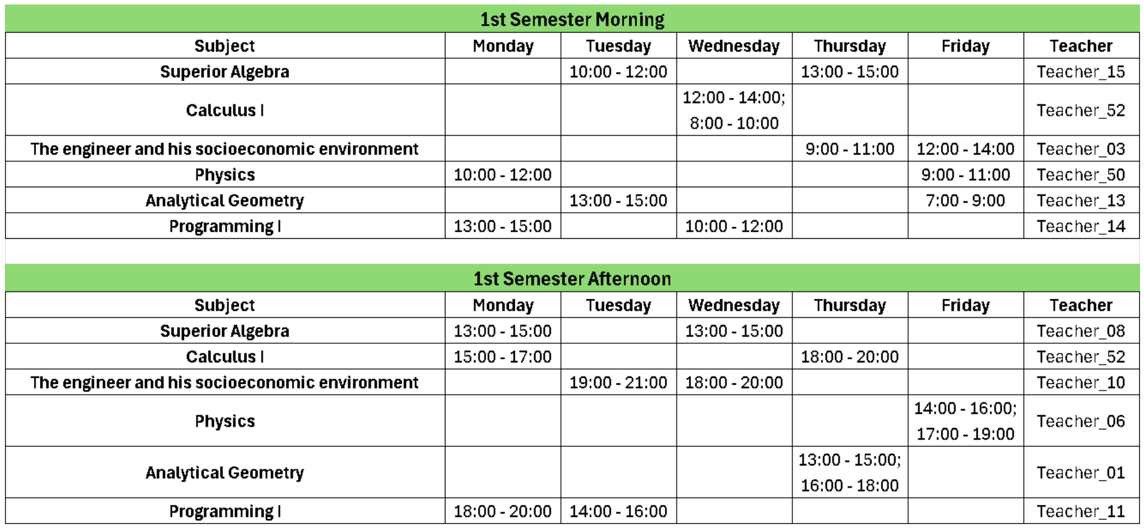


Fig. 9. Average performance in the ICO-CU-UAEM-VM dataset using the GA- GIRF

Fig. 9 shows the average behavior of the algorithm. The implementation of GA-GIRF made it possible to find a solution to the problem, particularly in the early generations. In some cases, although the algorithm initially failed to find a solution, the repair function enabled it to do so in later generations. Similarly, **Fig. 10** presents one of the schedules generated by GA-GIRF as an example. This schedule corresponds to one of the best individuals obtained during the runs and achieved an error value of zero.



1st Semester Morning

Subject	Monday	Tuesday	Wednesday	Thursday	Friday	Teacher
Superior Algebra		10:00 - 12:00		13:00 - 15:00		Teacher_15
Calculus I			12:00 - 14:00; 8:00 - 10:00			Teacher_52
The engineer and his socioeconomic environment				9:00 - 11:00	12:00 - 14:00	Teacher_03
Physics	10:00 - 12:00				9:00 - 11:00	Teacher_50
Analytical Geometry		13:00 - 15:00			7:00 - 9:00	Teacher_13
Programming I	13:00 - 15:00		10:00 - 12:00			Teacher_14

1st Semester Afternoon

Subject	Monday	Tuesday	Wednesday	Thursday	Friday	Teacher
Superior Algebra	13:00 - 15:00		13:00 - 15:00			Teacher_08
Calculus I	15:00 - 17:00			18:00 - 20:00		Teacher_52
The engineer and his socioeconomic environment		19:00 - 21:00	18:00 - 20:00			Teacher_10
Physics					14:00 - 16:00; 17:00 - 19:00	Teacher_06
Analytical Geometry				13:00 - 15:00; 16:00 - 18:00		Teacher_01
Programming I	18:00 - 20:00	14:00 - 16:00				Teacher_11

Fig. 10. Example of a Schedule Generated by GA-GIRF

Table 3, the value of the best individual found throughout each of the independent runs is presented, which allows an additional way to compare the results of both algorithms.

Table 3 Comparison of best results with S-GA and GA-GIRF.

	Mean	STD	Min	Max
S-GA	31.5	1.02	29	33
GA-GIRF	0.6	.66	0	2

6 Conclusion

This paper presented the most important aspects to consider when generating school timetables using evolutionary algorithms, comparing the implementation of a Simple Genetic Algorithm (S-GA) with a Genetic Algorithm featuring Guided Initialization and a Repair Function (GA-GIRF). The adaptability function was designed to evaluate whether all constraints were satisfied. A value of zero indicates that the generated schedule meets all the predefined conditions. Among the aspects evaluated are splicing of teachers across different groups and within the same group, teacher overload, overlapping classes, teacher availability, subject-teaching eligibility, and compliance with the required number of hours per subject per week.

Based on the experimental analysis and the hard constraints of the problem, the repair and initialization functions were implemented, successfully fulfilling the objectives and hypotheses of this work. Throughout the research, it was also observed that measuring population diversity—in terms of the similarity between individuals (i.e., class schedules)—is useful for understanding the behavior of the genetic algorithm. This approach can also aid in identifying and correcting errors that would otherwise be more difficult to detect.

Another noteworthy finding concerns the mutation operator. The use of restrictive mutation proved especially effective in scenarios with large teaching staffs, where each teacher is limited to teaching specific subjects based on their specialization. This mutation strategy helped guide the search toward solutions closer to the optimal. Finally, the comparison between GA-GIRF and S-GA using the ICO-CU-UAEM-VM dataset showed that GA-GIRF was capable of solving the scheduling problem, achieving a zero-error solution. This performance is largely attributed to the guided initialization, which generates schedules that are already close to optimal, and the repair function, which improves individuals over generations—enabling faster convergence, often within the early stages of evolution.

Based on the results obtained and the analysis conducted in this study, several directions for future work are proposed: a) Incorporation of Soft Constraints Evaluation: While the current adaptability function focuses on satisfying hard constraints, future work could extend the evaluation to include soft constraints—such as teacher preferences, balanced daily workloads, or avoiding back-to-back classes. Assigning weighted penalties to soft constraint violations may help generate more user-friendly schedules; b) Dynamic or Real-Time Scheduling: Exploring the application of the proposed approach in dynamic environments—where changes may occur in real time, such as teacher absences, room unavailability, or last-minute subject additions—could enhance the algorithm's robustness and practical utility and c) Development of a User Interface or Scheduling System: As a final application, a graphical interface or web-based system could be developed to allow institutional users to configure constraints and visualize the generated schedules, making the algorithm more accessible and practical for administrative use.

References

Abdelhalim, E. A., & El Khayat, G. A. (2016). *A utilization-based genetic algorithm for solving the university timetabling problem (UGA)*. *Alexandria Engineering Journal, 55*(2), 1395-1409. https://doi.org/10.1016/j.aej.2016.02.017

Abdullah, S., Turabieh, H., McCollum, B., & McMullan, P. (2012). *A hybrid metaheuristic approach to the university course timetabling problem.* Journal of Heuristics, 18(1), 1–23. https://doi.org/10.1007/s10732-010-9154-y

Abramson, D., & Abela, J. (1992). *A parallel genetic algorithm for solving the school timetabling problem.* En Proceedings of the Australian Computer Science Conference (pp. 1–11).

Assi, M., Halawi, B., & Haraty, R. A. (2018). *Genetic algorithm analysis using the graph coloring method for solving the university timetable problem*. *Procedia Computer Science, 126*, 899–906. https://doi.org/10.1016/j.procs.2018.08.024

Bejarano, G. (2011). Planificación de Horarios del Personal de Cirugía de un Hospital del Estado Aplicando Algoritmos Genéticos [Tesis]. Pontificia Universidad Católica de Perú.

Broca, J. M. (2016). *Algoritmos genéticos en la generación de horarios escolares* [Tesis de maestría, Universidad Autónoma del Estado de México]. Repositorio Institucional UAEMex. https://ri.uaemex.mx/handle/20.500.11799/63083

Castrillón, O. D. (2014). *Combinación entre algoritmos genéticos y aleatorios para la programación de horarios de clases basado en ritmos cognitivos*. *Información Tecnológica, 25*(4), 51–62. https://doi.org/10.4067/S0718-07642014000400008

Cortez Vásquez, A., Rosales Gerónimo, G., Naupari Quiroz, R., & Vega Huerta, H. Z. (2010). *Sistema de apoyo a la generación de horarios basado en algoritmos genéticos*. *Revista de Investigación de Sistemas e Informática, 7*(1), 37–55.

Flores, P., Brau, E., Monteverde, J. A., Salazar, N. F., Figueroa, J., Cadena, E., & Lizárraga, C. A. (2004). *Experimentos con algoritmos genéticos para resolver un problema real de programación maestros-horarios-cursos*. *Revista Iberoamericana de Sistemas, Cibernética e Informática, 1*, 42–46.

Guerra, M., Pardo, E., & Salas, E. (2013). *Problema del school timetabling y algoritmos genéticos: una revisión*. *Revista Vínculos, 10*(2), 259–276. https://doi.org/10.14483/2322939X.6478

Hafsa, M., Wattebled, P., Jacques, J., & Jourdan, L. (2023). *Solving a multiobjective professional timetabling problem using evolutionary algorithms at Mandarine Academy*. *International Transactions in Operational Research*. https://doi.org/10.1111/itor.13276

Jafari, H., & Salmasi, N. (2015). *Maximizing the nurses' preferences in nurse scheduling problem: Mathematical modeling and a meta-heuristic algorithm*. *Journal of Industrial Engineering International, 11*(3), 439–458. https://doi.org/10.1007/s40092-015-0111-0

Jha, S. K. (2014). *Exam timetabling problem using genetic algorithm*. *International Journal of Research in Engineering and Technology, 3*(5), 649–654.

Mahlous, A. R., & Mahlous, H. (2023). *Student timetabling genetic algorithm accounting for student preferences*. *PeerJ Computer Science, 9*, e1200. https://doi.org/10.7717/peerj-cs.1200

Savdekar, M., Bornare, C., Birari, K., & Kolhe, K. (2023). *Timetable generation system*. *International Research Journal of Modernization in Engineering Technology and Science, 5*(4), 1574–1577.

Shahvali Kohshori, M., & Saniee Abadeh, M. (2012). *Hybrid genetic algorithms for university course timetabling*. *International Journal of Computer Science Issues, 9*(2), 446.

Tan, Y. Y., Li, Y. X., & Wang, R. X. (2020). *Scheduling extra train paths into cyclic timetable based on the genetic algorithm*. *IEEE Access, 8*, 102199–102211. https://doi.org/10.1109/ACCESS.2020.2997777

Zandavi, S. M., Chung, V., & Anaissi, A. (2021). *Multi-user remote lab: Timetable scheduling using simplex nondominated sorting genetic algorithm*. *ACM/IMS Transactions on Data Science, 2*(2), 1–13. https://doi.org/10.1145/3437260