Exploring the Potential of Genetic Algorithms for Optimizing Academic Schedules at the School of Mechatronic Engineering: Preliminary Results

Johan Alarcón[1], Samantha Buitrón[1], Alexis Carrillo[1], Mateo Chuquimarca[1], Alexis Ortiz[1], Robinson Guachi[1], D. H. Peluffo-Ordóñez[2,3], and Lorena Guachi-Guachi[1,3](✉)

[1] Department of Mechatronics, Universidad Internacional del Ecuador, Av. Simon Bolivar, 170411 Quito, Ecuador
{joalarconag,sabuitronpa,alcarrilloro,machuquimarcaro, alortizsa,roguachigu,loguachigu}@uide.edu.ec

[2] College of Computing, Mohammed VI Polytechnic University, Lot 660, Hay Moulay Rachid, 43150 Ben Guerir, Morocco
peluffo.diego@um6p.ma

[3] SDAS Research Group, 43150 Ben Guerir, Morocco
{diego.peluffo,lorena.guachi}@sdas-group.com
https://sdas-group.com/

Abstract. The generation of schedules is a complex challenge, particularly in academic institutions aiming for equitable scheduling. The goal is to achieve fair and balanced schedules that meet the requirements of all parties involved, such as workload, class distribution, shifts, and other relevant criteria. To address this challenge, a genetic algorithm specifically designed for optimal schedule generation has been proposed as a solution. Adjusting genetic algorithm parameters impacts performance, and employing parameter optimization techniques effectively tackles this issue. This work introduces a genetic algorithm for optimal schedule generation, utilizing suitable encoding and operators, and evaluating quality through fitness techniques. Optimization efforts led to reduced execution time, improved solution quality, and positive outcomes like faster execution, fewer generations, increased stability, and convergence to optimal solutions.

Keywords: Equitable schedules · Genetic algorithm optimization · Resource allocation · Scheduling generation

1 Introduction

Scheduling generation is a complex challenge faced by various institutions and organizations in their day-to-day operations, including academic, professional fields, and other contexts where coordinating and allocating resources is necessary.

The problem lies in the need to find an appropriate combination of elements and constraints to meet the demands and needs of all involved parties. It is crucial to ensure that schedules are fair and equitable, avoiding situations where individuals or groups are disadvantaged in terms of workload, class distribution, shifts, or any other relevant criteria.

In the Mechatronics Engineering school at the International University of Ecuador, scheduling for courses and activities presents a complex challenge due to the variety of resources, constraints, and preferences of students and professors. Efficient schedule planning is crucial to ensure a fair distribution of subjects, avoid conflicts, and optimize resource allocation. To address this problem, the optimization of a genetic algorithm (GA) is proposed, aiming to find optimal or near-optimal solutions using techniques inspired by evolutionary computation.

Currently, various research studies are related to automatic schedule generation using GAs and other optimization techniques. These approaches have proven to be effective in solving resource allocation and scheduling problems in different academic contexts, which could be classified into two categories: classical genetic algorithms ([1–4]) and hybrid genetic algorithms ([5–7]).

A common problem found in the explored works is the adjustment of the involved parameters, such as population size, mutation rate, and the number of generations, which can significantly affect the performance of the algorithm for automatic schedule generation. To address this issue, techniques like parameter optimization can be employed to find the required parameter values.

The project aims to contribute to the generation of optimal schedules in the field of Mechatronics Engineering at the International University of Ecuador. To achieve this, the development of a GA is proposed, employing an appropriate encoding to represent schedules and using genetic operators such as selection, reproduction, and mutation to generate new solutions. Evaluation and fitness techniques will be applied to assess the quality of each solution, and multiple iterations of the algorithm will be performed to gradually improve the results. The goal is to obtain optimal or near-optimal schedules that meet all the established constraints and preferences, thereby enhancing the efficiency of course planning and resource allocation.

The obtained results so far support the feasibility and effectiveness of the proposed GA for schedule generation in the field of Mechatronics Engineering at the International University of Ecuador, resulting in reduced execution time and improved solution quality. The algorithm demonstrated positive outcomes, including lower average execution time, fewer required generations to find a solution, increased stability, and convergence towards optimal solutions. Our results are attributed to a higher number of elite schedules, increased mutation rate, and random class selection during mutation, leading to improved stability and reduced computational resources required.

This document is organized as follows: In Sect. 2, a brief overview of the current state of research is provided. Section 3 includes the description of the database used for experimentation, along with the definition of parameters and characteristic operators of the GA. The experimentation process and the metrics employed to evaluate the GA's performance are described in Sect. 4. The obtained results and the various comparisons conducted are presented in Sect. 5. Finally, Sect. 6 gathers the concluding remarks.

2 Related Works

The use of GAs in timetable generation has become an active and constantly evolving area of research, with numerous practical applications and possibilities for future development.

In the field of optimization, GAs have proven to be a powerful tool for solving complex problems. In particular, the generation of schedules is a task that can be solved using GAs and that has been the object of investigation in several studies. For instance, [1] and [8] described the implementation of a GA to generate university timetables with the goal of producing accurate solutions, although its implementation can be complicated and may require exhaustive tuning and testing. Authors in [9] described the application of a GA for teaching planning in a university, which, like the previous ones mentioned, can generate precise and efficient solutions, but its implementation may require a large amount of computing resources. Starting from the latter in terms of the limitation presented, it is similar to the one described in [6], with the difference that the work involved a hybrid GA to program the schedules of nurses in a hospital considering the nurse fatigue.

In the field of assigning academic timetables, various investigations have been carried out by using GAs. Among them are 'Assignment of academic schedules for the School of Civil Engineering in Computing of the University of Talca [10], which described the implementation of a GA to assign schedules in a school engineering. The advantage of this approach is that it can generate accurate solutions in a reasonable amount of time, although it may require adjustment to suit the specific needs of an engineering school. Other relevant articles presented in [11] and [12] described the planning of university timetables using GAs with an approach similar to [10], although with a complex implementation of the algorithm, which can require a large amount of computing resources as [9].

Authors in [3,13] reported the application of a GA to generate exam schedules in a university, with precise and optimal solutions, although its algorithm can be difficult to fit in. The work introduced in [13] presented the generation of schedules in a highschool with time efficient solutions, but with an algorithm that may require tuning and testing. Both works are relevant to this project, as they are part of the main literature to improve the selected algorithm.

In [5], the authors proposed a hybrid approach that combines a GA with a greedy approach to generate school and exam timetables efficiently but with a limitation on the greedy approach that can lead to sub-optimal solutions. Authors in [2,7] and [14] introduced an intelligent system based on a GA to plan schedules in various educational fields with optimal time solutions. With the limitation of being difficult to adjust the algorithm to adapt to the specific needs of the field to be applied. The GA introduced in [15] yield school timetables involving advanced technical knowledge.

Moreover, in the medical field, authors in [16] presented an appointment scheduling system for medical treatment using GAs presenting efficient solutions in terms of the use of time and medical resources, but using a large amount of information resources.

Based on the way of operating the works explored from the literature, we have identified two categories: 1) Classical GAs, which are capable of handling large search spaces and allow high-quality solutions to be found in a reasonable time, but can fall into local minima and do not always find the global optimal solution. Within this category we can find some articles such as [1,2] and [3]. 2) Hybrid GAs, which combine different techniques to improve the precision and efficiency in solving the problem, as well as they can improve the quality of the solutions found by classical genetic algorithms and reduce the time needed to find them, but their implementation may require more effort and technical knowledge than conventional GAs. Some of the articles categorized in this second point are [5,6] and [7].

3 Materials and Methods

3.1 Data

The database collects the information from seven semesters of the School of Mechatronics of the International University of Ecuador, as is shown in Table 1.

Table 1. Description of the data that will be used.

Data	Description	Value
Subjects	Subjects that need to use the classrooms	3 Subjects
Students per Semester	Number of students per semester	1st Semester: 25, 2nd Semester: 35, 3rd Semester: 25, 4th Semester: 30, 5th Semester: 35, 6th Semester: 45 and 7th Semester: 45
Classrooms	Maximum Capacity of students per classroom	A1: 25, A2: 45 and A3: 35
Professors	Professors who can impart the classes	4 Professors
Meeting Time	Times in which classes can be carried out	4 Meeting Times

Each of the chosen parameters are described in a general way, in [17] you can find the database in detail.

3.2 Proposed Algorithm

The proposed GA depicted in Fig. 1 was inspired by the work titled "Development of an exam scheduling system using genetic algorithm" [3], and aims at finding the best assignment of available subjects and resources.

Database: Within the database there are 5 academic objects: subjects, semesters, classrooms, teachers, and meeting schedules; these data will be imported into the GA as arrays.

Initialization of Population: A initial population of 9 random schedules based on the 5 aforementioned objects were created. This initial population served as a starting point for the search and optimization process that follows in the GA.

Fitness Calculation: The goal was to find the optimal schedule removing conflict with the objects that compose it. These conflicts can arise from conflicting reservations of professors or classrooms, classroom capacity, and professor availability. When the number of conflicts was zero, the problem was considered solved.

Parent Selection: If a schedule with zero conflicts was not found, parent selection by tournament was performed. This step selected randomly three schedules from the population in order to find the two best parents for the next generation.

Crossover: Each class in the new schedule, made up of the 5 aforementioned objects, was crossed with a randomly selected class from the parent schedules, thus producing reproduction through a crossing point.

Mutation: For schedules in the population that are not part of the elite, random classes are mutated with a probability determined by the mutation rate. If a mutation occurs, the selected class is replaced with a random class from another schedule generated by the population initialization method.

Elitism: This ensured that the best fitness schedule obtained so far was kept in the population and transmitted to future generations without changes, avoiding the loss of high-quality solutions.

3.3 Changes Applied to [3]

We conducted a set of experiments to improve the overall performance of the work proposed in [3], which involved adjustments to the elitism parameter and mutation rate, as well as a modification to the mutation operator technique.

The number of elite schedules was increased from 1 to 2. This means that the two best schedules from each generation will be preserved in the next generation. The mutation rate was increased from 0.1 to 0.2 aiming at promoting greater solution diversity.

Regarding the mutation technique, the probabilistic approach to determining which classes will be mutated was maintained. However, a change was introduced in the mutation process. In [3], mutation was performed sequentially, that

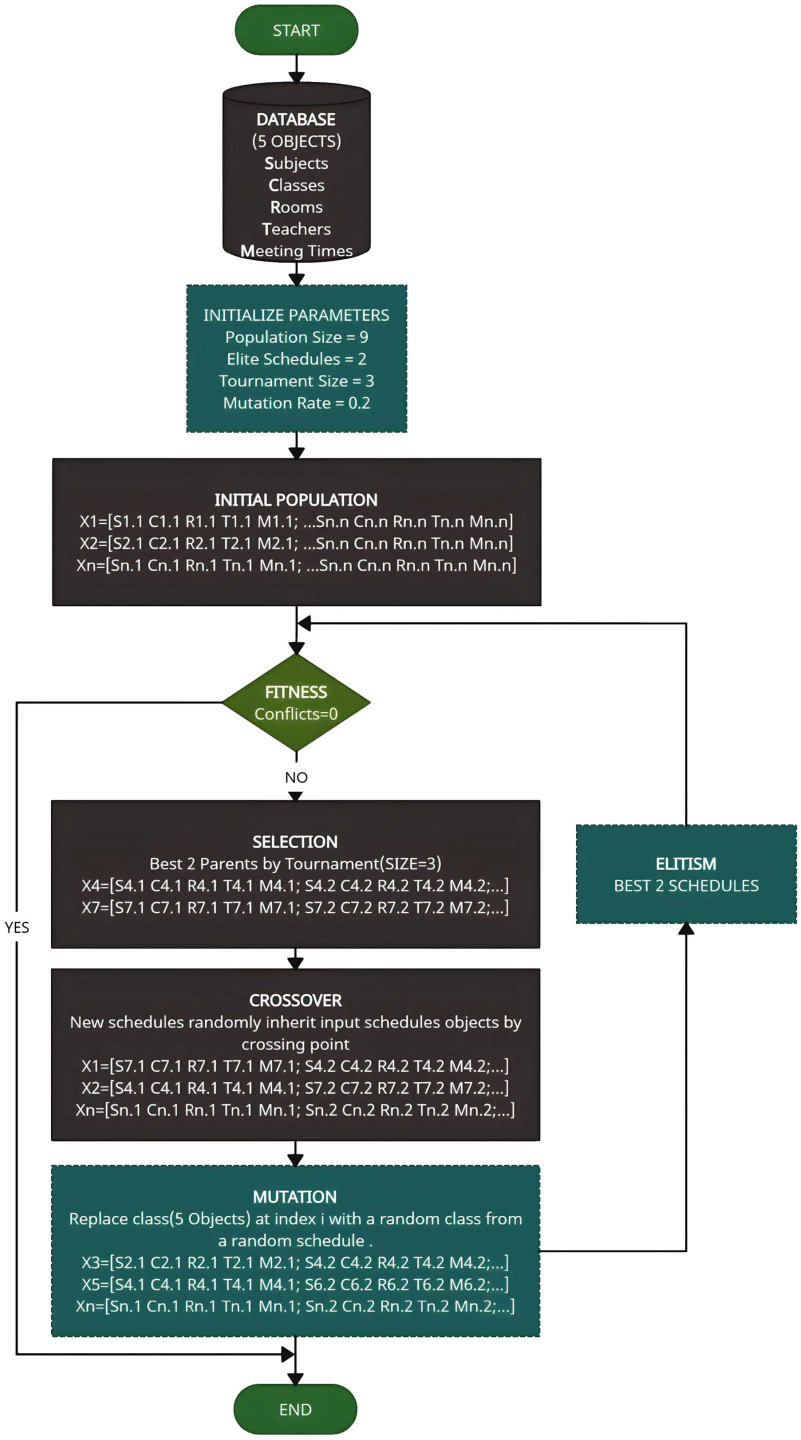

START
DATABASE
(5 OBJECTS)
Subjects
Classes
Rooms
Teachers
Meeting Times
INITIALIZE PARAMETERS
Population Size = 9
Elite Schedules = 2
Tournament Size = 3
Mutation Rate = 0.2
INITIAL POPULATION
X1=[S1.1 C1.1 R1.1 T1.1 M1.1; ...Sn.n Cn.n Rn.n Tn.n Mn.n]
X2=[S2.1 C2.1 R2.1 T2.1 M2.1; ...Sn.n Cn.n Rn.n Tn.n Mn.n]
Xn=[Sn.1 Cn.1 Rn.1 Tn.1 Mn.1; ...Sn.n Cn.n Rn.n Tn.n Mn.n]
FITNESS
Conflicts=0
NO
YES
SELECTION
Best 2 Parents by Tournament(SIZE=3)
X4=[S4.1 C4.1 R4.1 T4.1 M4.1; S4.2 C4.2 R4.2 T4.2 M4.2;...]
X7=[S7.1 C7.1 R7.1 T7.1 M7.1; S7.2 C7.2 R7.2 T7.2 M7.2;...]
ELITISM
BEST 2 SCHEDULES
CROSSOVER
New schedules randomly inherit input schedules objects by crossing point
X1=[S7.1 C7.1 R7.1 T7.1 M7.1; S4.2 C4.2 R4.2 T4.2 M4.2;...]
X2=[S4.1 C4.1 R4.1 T4.1 M4.1; S7.2 C7.2 R7.2 T7.2 M7.2;...]
Xn=[Sn.1 Cn.1 Rn.1 Tn.1 Mn.1; Sn.2 Cn.2 Rn.2 Tn.2 Mn.2;...]
MUTATION
Replace class(5 Objects) at index i with a random class from a random schedule .
X3=[S2.1 C2.1 R2.1 T2.1 M2.1; S4.2 C4.2 R4.2 T4.2 M4.2;...]
X5=[S4.1 C4.1 R4.1 T4.1 M4.1; S6.2 C6.2 R6.2 T6.2 M6.2;...]
Xn=[Sn.1 Cn.1 Rn.1 Tn.1 Mn.1; Sn.2 Cn.2 Rn.2 Tn.2 Mn.2;...]
END

Fig. 1. Flowchart of the proposed genetic algorithm to generate the academic schedules.

is, class by class. In this work, we maintained the ordered sequence of the algorithm that will be mutated, but a random position within the schedule to be introduced was chosen. With these changes, it was expected to explore different schedule combinations and increased solution diversity in the search for improved optimization.

4 Experimental Setup

For experimental purposes, the hardware resources involved an 8-core AMD Ryzen 7 5800X processor, 16 GB of RAM, and an LG 20MP38HQ-B monitor. The software resources involved Windows 10 Pro x64-bit, Python in Google Colab, and Opera web browser. The algorithm was implemented through Python software routines. In the development of the proposed algorithm, the following libraries were employed: time, math, sqlite3, prettytable, random, and enum.

4.1 Metrics

- **Generations:** The number of generations required to find the solution to the problem.
- **Execution Time:** The execution time is an important indicator of the algorithm's efficiency in obtaining high-quality schedules within a reasonable time.
- **Overall Entropy:** An algorithm that tends to generate similar solutions in each iteration is less desirable than one that produces diverse solutions. The diversity of the population was measured using the population's entropy, represented by (1).

  $$H = -\sum(\rho \cdot \log_2(\rho)) \tag{1}$$

  where:
  ρ proportion of each type of solution in the population;
  $\log_2$ logarithm base 2.
  The values of H can range from 0 to $\log_2(N)$, where N is the size of the population. A value of H close to 0 indicates a homogeneous population, while high entropy indicates a more diverse population.
- **Overall Convergence:** Convergence refers to how quickly the genetic algorithm finds an optimal solution or one close to it. The convergence speed was measured by the fitness improvement rate represented by (2).

$$\textit{Fitness improvement rate} = \frac{\textit{current generation avg. fitness} - \textit{avg. fitness of the previous generation}}{\textit{avg. fitness of the previous generation}} \tag{2}$$

- **Average Fitness:** It measures how well a schedule meets the established requirements and constraints, i.e., its quality. In this case, the average fitness of each generation was calculated, and then an overall average was obtained.

4.2 Parameters of the Evolutionary Algorithm

Table 2 includes the comprehensive set of parameters utilized in our algorithm. These parameters hold significant importance in driving the optimization process effectively. The parameter values were carefully established experimentally and were also drawn from literature as shown the column value selection protocol in Table 2.

Table 2. Parameters values used in the proposed algorithm.

Name	Description	Value	Value Selection Procedure
POPULATION_SIZE	Size of population in each generation	9	Based on the problem and available resources. Larger populations can explore more search space but require more computation [3]
NUMB_OF_ELITE_SCHEDULES	The number of elite schedules that are preserved without changes in each generation	1	Based on the problem and experimentation
TOURNAMENT_SELECTION_SIZE	The size of the tournament selection pool	3	Based on the problem and experimentation
MUTATION_RATE	The probability of a gene in an individual being mutated	0.1	Based on the problem and experimentation
VERBOSE_FLAG	Flag to control the output of detailed information during the execution of the algorithm	False	Based on the user's preference

The code is available at the following link: source code.

5 Experimental Results

5.1 The Proposed Algorithm vs. [3]

The increase to two elite programs permitted our algorithm to retain high-quality solutions from one generation to the next one, accelerating the convergence of the algorithm. Although there could have been an effect on the population entropy due to the repetition of solutions, this problem did not arise.

The increase in the mutation rate, specifically from 0.1 to 0.2 permitted our algorithm to explore more solutions. It is important to note that a high mutation rate can make it difficult to find the solution and generate less stable or suboptimal solutions. However, in this particular case, this problem was not experienced.

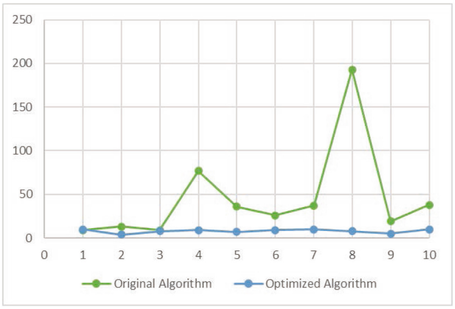
The change applied to the mutation operator favored greater diversity and increased the chances of finding optimal solutions. Although there is a possibility of generating less coherent schedules or schedules that do not meet certain restrictions, it was established that the random position should be within the allowed values in the schedule, thus avoiding this situation.

In summary, the changes proposed in our algorithm provided benefits such as faster convergence, wider exploration of solutions and greater diversity, without generating significant problems in solution quality or schedule coherence.

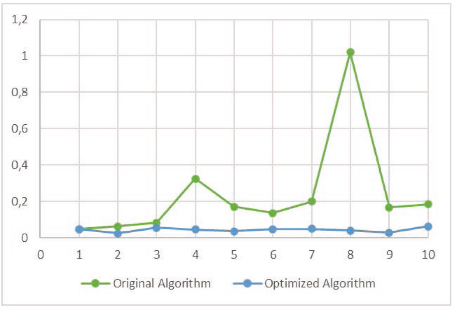
Table 3 and 4 depict the results obtained from 10 iterations of each algorithm. In order to facilitate the comprehension of the outcomes for each scenario, Fig. 2 provides a visual comparison of both algorithms across various metrics.

Table 3. Overall performance of the algorithm presented in [3] through 10 executions.

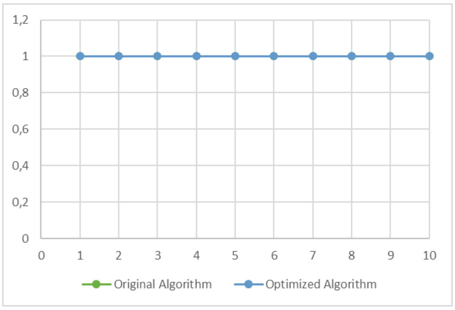
Execution	Generations	Execution Time [s]	Overall Entropy	Overall Convergence	Average Fitness
1	9	0,048	1	1,235	0,366
2	13	0,062	1	1,377	0,379
3	9	0,081	1	1,230	0,273
4	77	0,325	1	1,614	0,375
5	36	0,170	1	1,473	0,350
6	26	0,136	1	1,606	0,368
7	37	0,200	1	1,458	0,318
8	193	1,021	1	2,501	0,370
9	19	0,167	1	1,498	0,364
10	38	0,183	1	1,545	0,334
Average	46	0,239	1	1,554	0,350



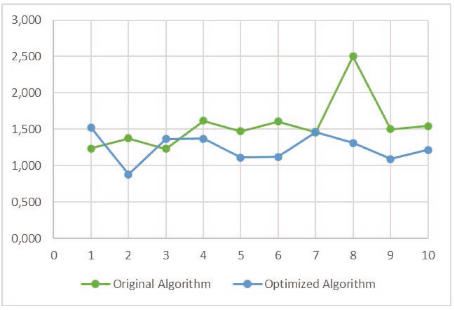
(a) Number of Generations



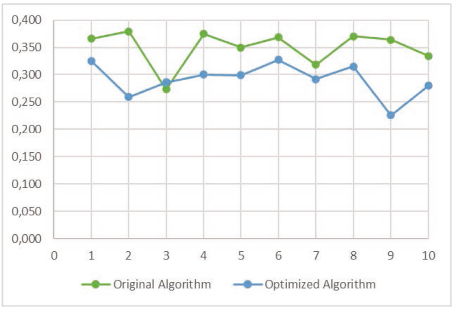
(b) Execution Time



(c) Overall Entropy



(d) Overall Convergence



(e) Average Fitness

Fig. 2. Overall performance of the proposed algorithm in comparison with [3].

Table 4. Overall performance of the proposed algorithm through ten executions.

Execution	Generations	Execution Time [s]	Overall Entropy	Overall Convergence	Average Fitness
1	10	0,047	1	1,525	0,325
2	4	0,024	1	0,880	0,259
3	8	0,054	1	1,367	0,286
4	9	0,046	1	1,368	0,300
5	7	0,036	1	1,112	0,299
6	9	0,047	1	1,120	0,327
7	10	0,049	1	1,461	0,292
8	8	0,040	1	1,311	0,315
9	5	0,028	1	1,093	0,226
10	10	0,063	1	1,215	0,280
Average	8	0,043	1	1,245	0,291

5.2 Discussion

From the obtained results Fig. 2 and Table 4, it can be observed that the proposed algorithm achieved a significant improvement in terms of the evaluated metrics. Firstly, it reduced the average algorithm execution time, which led to a decrease in the average number of generations required to find the solution. In [3], a maximum of 193 generations and a minimum of 9 generations were required, while in our algorithm, these values were reduced to a maximum of 10 generations and a minimum of 4 generations as can be seen in Table 3 and 4.

In addition, the obtained results depicted in show greater stability in the search for the solution in the optimized algorithm compared to the original as can be seen in Fig. 2. In the original algorithm, there were cases where the results were considerably dispersed, while in the optimized algorithm, this dispersion was significantly reduced.

Regarding entropy, it remained at the same level as in the original algorithm since it was already at its maximum value as shown in Fig. 2c. Therefore, the changes made did not affect this aspect.

In terms of the overall convergence of the algorithm, the optimized algorithm has achieved similar convergence to the original algorithm but in fewer generations as seen in the Fig. 2d. This means that the solutions obtained in each generation were optimal and approached closer to the final solution as the execution of the algorithm progressed.

In both algorithms, a very good average fitness has been obtained with an almost linear trend Fig. 2e. This indicates that the algorithm has been effective in improving solutions generation after generation.

In summary, the experiment has demonstrated an improvement in evaluated metrics such as execution time, number of required generations, result in stability, and quality of obtained solutions. These results support the effectiveness of changes implemented in genetic algorithms.

6 Conclusion

In conclusion, this work aimed to optimize a GA for scheduling, seeking to reduce execution time and improve the quality of obtained solutions. The results obtained were very positive, showing a clear improvement in evaluated metrics. A significant reduction was achieved in the average execution time and the number of generations required to find the solution, as well as greater stability and convergence towards optimal solutions in fewer generations. The main advantage of the proposed algorithm lies in the combination of a higher number of elite schedules preserved in each generation, a higher mutation rate, and a random selection of classes during mutation, achieving greater stability and reduction of computational resources compared to the original algorithm. This allows a greater diversity of solutions and a more efficient search. However, a possible disadvantage of the algorithm could be its lack of generalizability to other databases since the algorithm would have to be adapted to the constraints of the new data which may imply changes in the implemented code.

As a future improvement, it is suggested to consider incorporating crossover operators into the proposed genetic algorithm. This would allow combining features of elite schedules and exploring new solutions, which could further increase diversity and quality of obtained solutions. Overall, the obtained results and possible future improvements highlight the effectiveness and potential of the proposed genetic algorithm in scheduling.

References

1. Kakkar, M.K., Singla, J., Garg, N., Gupta, G., Srivastava, P., Kumar, A.: Class schedule generation using evolutionary algorithms. J. Phys. Conf. Ser. **1950**(1), 012067. IOP Publishing (2021)
2. Bimantara, I., Yuhana, U.L., Supriana, I.W., Pardede, E.: An intelligent system based on evolutionary algorithm for scheduling university course timetable. Wayan and Pardede, Eric, An Intelligent System Based on Evolutionary Algorithm for Scheduling University Course Timetable
3. Adesagba, O.E.: Development of an examination timetabling system using genetic algorithm (2021)
4. Fuenmayor, R., et al.: A genetic algorithm for scheduling laboratory rooms: a case study. In: Florez, H., Gomez, H. (eds.) Applied Informatics. ICAI 2022. CCIS, vol. 1643, pp. 3–14. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-19647-8_1
5. Prosad, R., Khan, M., Rahman, A., Ahammad, I.: Design of class routine and exam hall invigilation system based on genetic algorithm and greedy approach. Asian J. Res. Comput. Sci. **13**(3), 28–44 (2022)
6. Amindoust, A., Asadpour, M., Shirmohammadi, S.: A hybrid genetic algorithm for nurse scheduling problem considering the fatigue factor. J. Healthc. Eng. **2021** (2021)

7. Terán-Pozo, E.E., Romero-Fernández, A.J., Sandoval-Pillajo, A.L., Freire-Lescano, L.R.: Influencia de los algoritmos genéticos en la generación de horarios en unidad educativa. CIENCIAMATRIA **8**(4), 876–891 (2022)
8. Xu, J.: Improved genetic algorithm to solve the scheduling problem of college English courses. Complexity **2021**, 1–11 (2021)
9. Henry Nelson, A., Fuentes, F.J.A., Candelaria, M.R.H.: La planificación docente utilizando algoritmos genéticos. Revista Didasc@ lia: Didáctica y Educación **12**(4) (2021)
10. Gálvez Toledo, Y.A., et al.: Asignación de horarios académicos para la escuela de ingeniería civil en computación de la universidad de talca utilizando algoritmos genéticos, Ph.D. dissertation, Universidad de Talca (Chile). Escuela de Ingeniería Civil en Computación (2021)
11. Gomez, E.F.: Programación de horarios universitarios jerárquicos 2019 (2021)
12. Contreras, L.A.C.: Búsqueda de soluciones factibles para el problema de horarios de cursos universitarios (2022)
13. Pitoňáková, K.: Class schedule generator
14. Nugroho, A.K., Permadi, I., Yasifa, A.R., et al.: Optimizing course scheduling faculty of engineering unsoed using genetic algorithms. JITK (Jurnal Ilmu Pengetahuan dan Teknologi Komputer) **7**(2), 91–98 (2022)
15. Acuña-Galván, I., Lezama-León, E., Bolaños-Rodríguez, E., Solís-Galindo, A.E., Vega-Cano, G.Y.: Generación de horarios mediante algoritmos genéticos. Boletín Científico INVESTIGIUM de la Escuela Superior de Tizayuca **8**(Especial), 51–57 (2022)
16. Suresh, K., Joseph, B., et al.: Patient scheduling system for medical treatment using genetic algorithm. J. Popul. Ther. Clin. Pharmacol. **30**(8), 268–273 (2023)
17. Base de datos. https://n9.cl/f5vy6