NOVA IMS

Information Management School

TIMETABLE SCHEDULING PROBLEM

Project developed by:

Daniel Branco, 20220599

Inês Ventura, 20220612

Maria Mendonça, 20220625

Miguel David, 20220622

https://github.com/mariaMendonca1/TimetableScheduling.git

1. The problem

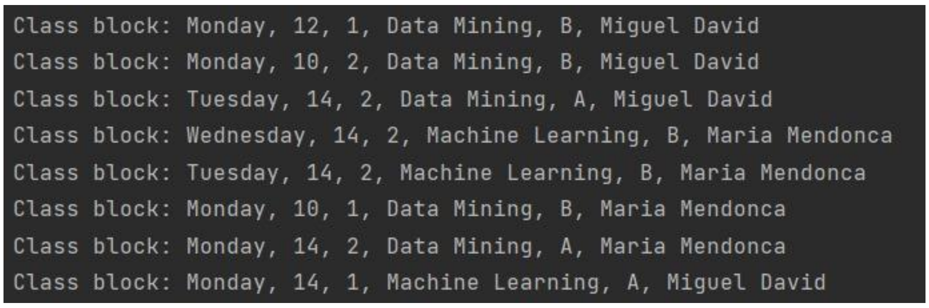
This report focuses on explaining the implementation of a timetable scheduling optimization problem in a university domain. The objective is to address the challenges associated with creating efficient and effective schedules that meet students' needs. Therefore, as students ourselves, this topic is of utmost importance for all team members.

The adopted approach involved a progressive methodology. It started by utilizing simple data and straightforward examples to obtain optimal solution schedules. Subsequently, the complexity was incrementally increased by incorporating more intricate datasets and imposing additional constraints.

Throughout this paper, it will be mentioned in detail the representation chosen, all fitness functions experimented, along with selection methods, crossover and mutation operators used. Lastly, relevant findings and results will be compared and discussed.

1.1. Representation

Initially, the group faced the task of determining relevant attributes of each lecture that should be taken into account. For every class block, it was considered information about the day of the week, timeslot, classroom, subject, and group. Professors were later included. These attributes formed the essential arguments for constructing a *class* representing each lecture. Consequently, each chromosome was represented as a list of these *classes*, encapsulating the necessary information to build an entire timetable.



```
Class block: Monday, 12, 1, Data Mining, B, Miguel David
Class block: Monday, 10, 2, Data Mining, B, Miguel David
Class block: Tuesday, 14, 2, Data Mining, A, Miguel David
Class block: Wednesday, 14, 2, Machine Learning, B, Maria Mendonca
Class block: Tuesday, 14, 2, Machine Learning, B, Maria Mendonca
Class block: Monday, 10, 1, Data Mining, B, Maria Mendonca
Class block: Monday, 14, 2, Data Mining, A, Maria Mendonca
Class block: Monday, 14, 1, Machine Learning, A, Miguel David
```

Figure 1: Example of individual

The chosen representation offers several compelling reasons for its effectiveness. Firstly, its clear and concise structure simplifies the understanding of the data. By representing each lecture as an individual entity, there is more flexibility in manipulating and rearranging the classes. By using a list of classes, the representation can scale effectively, which facilitates handling with larger datasets. It also guarantees the same length for every possible individual, which is the ultimate goal.

Due to the complexity of the scheduling problem, binary encoding or similar strategies were not employed as they proved unsuitable, to some extent, for effectively representing the diverse attributes and constraints involved. For instance, the need to accommodate multiple attributes, such to express all possible different options for each attribute, would require lengthy binary codes, resulting in excessively long individuals.

The suggested representation implies that the size of each individual is dependent on the number of groups, subjects, and the number of weekly lectures per subject. It is important to mention that the duration of all lectures was assumed to be two hours, and all groups are enrolled in the same program, implying that they attend all available subjects.

1.2. Fitness Function

Finding an optimal timetable is a minimization problem in the sense that the primary objective is to minimize the number of overlapping classes. The fitness functions designed for this purpose count the number of conflicts among all classes and the goal is to find a timetable that achieves a fitness value of 0.

For the initial and simplest fitness function created (*fitness1*), a conflict was assigned whenever the following conditions were met:

- A classroom was simultaneously occupied by two different lectures.
- A group had overlapping lectures, attending two different classes at the same time.
- A group had the same lecture scheduled twice in a day.
- A group had an incorrect number of lectures per subject per week, either less or more than the two required (+1 conflict only for any value different than two).

If, in a timetable, two blocks of classes are exactly the same, it is important to consider the conflicts mentioned above, rather than just counting it as a single conflict.

The second fitness function (*fitness2*) takes into account additional constraints, considering the needs and well-being of the students. The constraints are the following:

- A conflict occurs if a timetable does not have at least one free day.
- A conflict arises if, within a single day, a group is required to change classrooms for different lectures.
- A conflict exists if the schedule for each day exceeds the maximum number of classes permitted per day.

Once professors were introduced, a set of additional constraints emerged, as outlined below (*fitness3*):

- Each teacher is limited to a maximum of three classes per day.
- A professor cannot simultaneously teach two lectures.
- A professor is associated with a specific subject, representing his/her/their area of expertise. Consequently, whenever a professor is assigned to teach other lectures, a conflict should be considered.

The process of enhancing the fitness function followed a similar approach to that of expanding the data used for the problem. This involved a step-by-step procedure, starting with simple data (*simple_data*) and gradually escalating to a larger dataset (*complex_data*). Meanwhile, the number of attributes associated with each class was also expanded, with the addiction of teachers. By adopting this approach, the group aimed to assess the robustness of the proposed solution and to align it more closely with real-life scenarios.

2. Selection Methods

For this project, it was followed a similar approach to the lessons conducted throughout the semester by using tournament selection, fitness proportionate selection, and rank-based selection. However, in the Findings and Results section, the group made the decision to focus solely on tournament selection. This choice was primarily influenced by the flexibility provided by tournament selection, as it allows to fine-tune the selection pressure through the adjustment of a single parameter: the tournament size. For this problem, it was set at 4.

3. Crossover operators

Regarding crossover operators, single-point crossover, uniform crossover and attribute-based crossover were implemented. This section provides a brief explanation of the two operators that were not covered during the course.

Uniform crossover consists of randomly selecting each gene from either parent with equal probability.

The team developed one alternative crossover operator, named attribute-based crossover. It involves selecting a random number of changes in attributes, between $\frac{1}{4}$ and $\frac{3}{4}$ of the total attributes. Once an attribute and a class index from the entire timetable are randomly selected, the crossover must be executed for that attribute in the corresponding position. The procedure is repeated until the number of attributes that were initially randomly chosen have undergone crossover. This method prevents the selection of attributes that have already been chosen.

Partially matched crossover and cycle crossover were deemed unsuitable for this specific example due to their reliance on unordered indexes for both parents. These crossover operators cannot be implemented with the chosen representation, as altering the order of each class of the chromosome does not impact the outcome.

4. Mutation Operators

Swap mutation, inversion mutation and binary mutation were not applicable in this context. Swap mutation and inversion mutation rely on the premise that changing the gene order results in a different chromosome. However, for the chosen timetable representation, the order of the class blocks does not impact the solution, as mentioned earlier. Binary mutation, as the name suggests, requires binary representations.

For this reason, some operators were created to meet the group's necessities. Single point mutation randomly selects one gene and changes one of its attributes. Furthermore, the attribute exchange mutation arbitrarily selects two classes from the given timetable and exchanges half of their attributes with each other. Finally, the new gene mutation operates by randomly generating a class index. This generated index is then used to conduct a swap between the original gene and a newly randomly created class.

5. Findings and results

Unlike the selection method, an evaluation of all possible configurations of different crossover and mutation operators was conducted to identify the most effective pair to address the given problem. Similarly to choosing right away tournament selection, elitism was implemented in all attempts, as it has consistently improved the results. Elitism was used to ensure that the best-performing individuals were preserved.

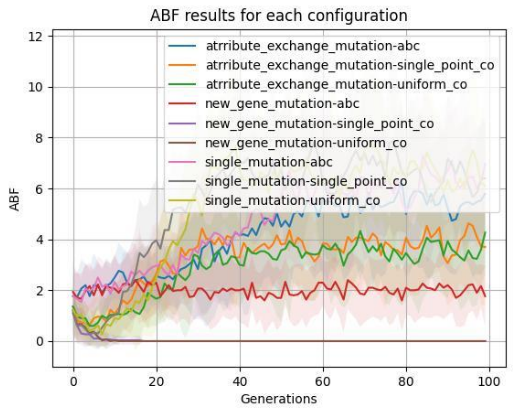


Figure 2: Simple Data and Fitness1 without implementing elitism.

In total, 9 configurations were tested, and their performances analyzed. For comparison purposes, all configurations were executed using the three different fitness functions developed and both datasets. Each test consisted of 30 runs per configuration, 100 generations with population size equal to 100. The probabilities for mutation and crossover were set at 0.05 and 0.9, respectively. This configuration has consistently demonstrated the ability to achieve optimal results, which was evidenced in the specific example at hand.

All configurations for every dataset and fitness function, with respective illustrations also shown below, were saved in separate folders for simplicity purposes.

Simple Data

Complex Data

Fitness 1

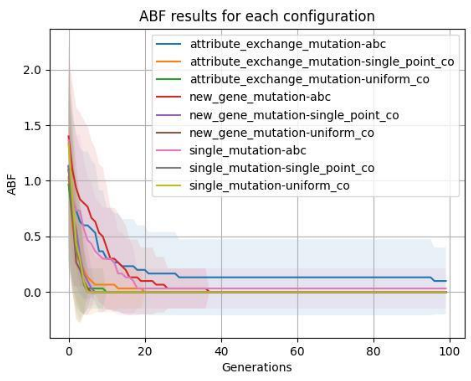


Figure 3

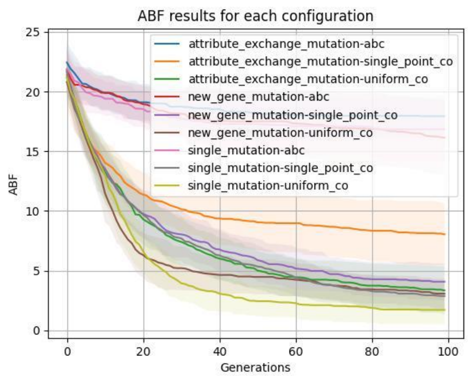


Figure 4

Fitness 2

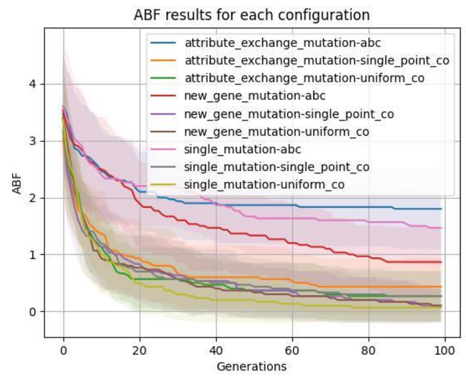


Figure 5

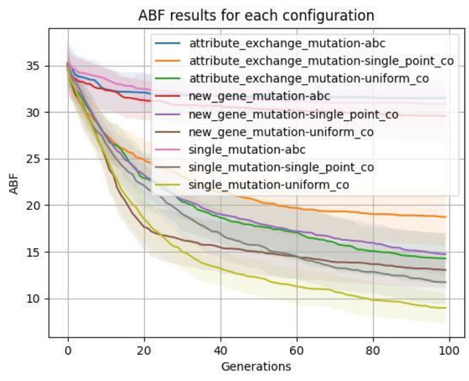


Figure 6

Fitness 3

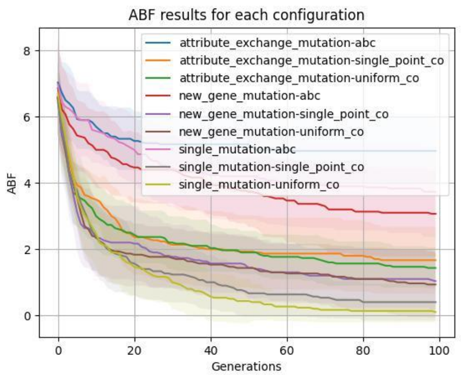


Figure 7

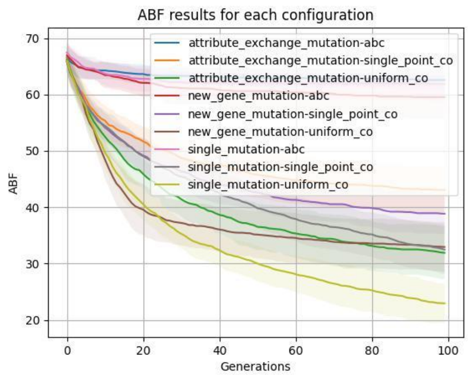


Figure 8

The optimal configurations were identified by plotting all potential setups for each fitness function and dataset and analyzing the average of the highest fitness value achieved per generation and its corresponding standard deviation.

Upon reviewing the results presented above, it becomes evident that the most effective combinations of crossover and mutation operators are uniform crossover along with single mutation, as well as uniform crossover paired with new gene mutation. In contrast, the poorest performing configurations consistently include the attribute-based crossover operator, since it exhibits minimal evolutionary progress.

The fitness functions elaborated impact the convergence of the genetic algorithm. As observed in the left column of the aforementioned table, it becomes clear that as the complexity of the fitness function increases, *ceteris paribus*, the number of generations required to reach optimal fitness values also increases. This is evident from the slower rate at which the curves' slopes decline.

As anticipated, using the complex dataset brought a greater challenge in achieving a fitness value of 0. Consequently, the team opted to modify certain hyperparameters to assess their impact on the obtained results when using the complex dataset. The parameters under investigation included the crossover and mutation rates, population size, and the number of runs conducted for each configuration.

First, the group decided to double the population size to enable a more extensive exploration of the solution space and increase the chances of retaining diverse individuals with different features. However, the outcomes did not show significant improvements in fitness values, except when using the fitness1 function. Realizing that population size alone was not sufficient to address the issues at hand, the focus shifted to adjusting the mutation and crossover rates. The mutation rate was increased to 0.2, while the crossover rate was decreased to 0.8, pushing the values towards the upper and lower limits commonly used for each. Despite these modifications, no significant improvements were observed for both fitness2 and fitness3. To further optimize the algorithm, the number of runs was increased to 40. Given that changes in hyperparameters can have varying effects when combined together or applied individually, all the aforementioned alterations were mixed together. Once again, only when using the fitness1 function did the algorithm come close to achieving a fitness value of 0. The resulting graphs of these experiments were saved in the "Results/Improving" directory.

6. Conclusion

The group is satisfied with the achieved outcomes. The primary objective of identifying the optimal selection-crossover-mutation combination was accomplished, with a fitness function of 0 being attained for most cases. Furthermore, the implementation and comprehension of these operators played a crucial role in enhancing the understanding of the field for all team members.

However, it is worth mentioning potential areas for improvement. Given unlimited time, the group would have conducted a larger number of runs, with an increased number of generations for each configuration and fitness function. This approach could potentially yield better results, particularly for the complex dataset when using *fitness2* and *fitness3* functions.

The workload was distributed evenly among all team members, with collaborative efforts in the conceptualization and execution of fitness functions, representations, and all relevant implementations. These tasks were thoroughly discussed and deliberated during meetings to ensure consensus and obtain collective approval from all team members.