MDSAA

Master Degree Program in

Data Science and Advanced Analytics

Intelligence Computation for Optimization

Marisa Sequeira 20230989

Hugo Oliveira 20230745

Diogo Silva 20201643

Group 'Bacalhau com datas'

https://github.com/diogovs09/CIFO.git

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

May, 20

Index

Introduction

Organizing and planning are essential responsibilities in various settings, such as educational institutions. But these tasks can get complicated and tough to handle with regular planning methods. This is where Genetic Algorithms (GAs) can be useful. GAs are smart search techniques inspired by natural selection and genetics great for finding good solutions in large and complex search spaces, making them perfect for solving many issues, including scheduling problems.

In this project, we're using GAs to tackle the University Course Timetabling problem. The goal is to create a timetable that meets all the necessary constraints and to show how effective these algorithms can be in solving real-world challenges.

Problem Representation

The problem of creating an optimal university course timetable involves accounting for various constraints and attributes associated with each class. To address this, was adopted a representation that encapsulates all these relevant attributes within a Class object, including attributes for the day, timeslot, classroom, subject, teacher, and group. These `Class` objects are then aggregated into an `Individual` object, organized as a list.

This representation offers several advantages. Firstly, encapsulating each lecture as a 'Class' object enhances data comprehension and manipulation, promoting modular and flexible structuring. This allows for straightforward manipulation and rearrangement of classes. Secondly, utilizing a list of 'Class' objects ensures scalability and consistency, efficiently handling larger datasets without compromising manageability. Additionally, the uniform length of timetables as lists simplifies genetic algorithm operations, preserving structural integrity effectively.

The chosen 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 also important to mention that the duration of all lectures was assumed to be one hour, and all groups are enrolled in the same program, implying that they attend all available subjects.

It's worth emphasizing that binary encoding or similar strategies were deemed inappropriate for this problem due to the complexity and diverse nature of the attributes involved. Representing the multiple attributes of each class using binary codes would result in excessively long and unwieldy individuals, complicating not only the implementation but also hinder the performance of the GA due to the increased computational overhead.

Fitness Function Design

The primary objective of optimizing the university course timetable is to create an efficient schedule that minimizes scheduling conflicts and maximizes resource utilization, with fitter individuals being more likely to contribute to the next generation. To achieve this, fitness functions were designed to identify these conflicts, minimize them, and attain an optimal schedule with a fitness value of 0. Thus, the optimization task manifested itself as a minimization problem.

We decided to follow a progressive approach in enhancing the fitness functions. This method involved starting with simple constraints and gradually incorporating more complex scenarios. By adopting this strategy, we aimed to evaluate the robustness of the solution and iteratively optimize the university course timetable.

Three distinct fitness functions were developed, each assigning conflicts under specific conditions:

- get_fitness1: This initial function focuses on basic conflicts, checking for overlapping lectures within the same group, classroom double bookings, duplicate lectures on the same day for a group, and incorrect weekly lecture counts per subject. These basic checks were essential for identifying and resolving fundamental scheduling issues.

- get_fitness2: This version incorporates extended checks to ensure the well-being of students. It ensures that each group has at least one free day, prevents classroom changes within a single day, and limits the number of classes per group, per day to three.

- get_fitness3: The most advanced fitness function, incorporating all previous checks, introduces teacher-specific constraints. It ensures that no teacher is overloaded with more than three classes per day, prevents teachers from teaching two lectures simultaneously, and matches each teacher with their designated subjects.

Genetic Operators

Various genetic operators inspired by those introduced in practical lessons during the semester were adopted. The inclusion of these operators serves the purpose of exploring different combinations within the algorithm to identify the most effective configuration of selection, crossover, and mutation methods. Each combination will undergo evaluation based on its performance, ensuring a comprehensive examination of the potential advantages and limitations of each operator in the context of our problem.

- <u>Selection Methods</u>

Two distinct selection methods were implemented: fitness proportionate selection (FPS) and tournament selection. Given its simplicity, it is expected that tournament selection will exhibit greater computational efficiency compared to FPS.

- <u>Crossover Operators</u>

In our genetic algorithm implementation, we integrated single-point crossover, cycle crossover, and introduced two novel operators: uniform crossover and attribute-based crossover (ABC). Single-point crossover and cycle crossover proved unsuitable for this scenario due to their reliance on unordered indexes for both parents. Since altering the order of classes in the chromosome does not impact the outcome in this representation, these crossover methods cannot be effectively implemented.

Here's a brief overview of the new crossover operators, not covered during the course:

• uniform crossover: It randomly selects genes from each parent with equal probability to generate offspring.

• Attribute-Based Crossover (abc): It operates by exchanging attributes between objects. It involves randomly selecting a subset of attributes, ranging from 1/4 to 3/4 of all attributes in the timetable, and do the crossover for that attribute in the corresponding positi. This process continues until the specified number of attribute changes is achieved, ensuring that previously selected attributes are not chosen again. The goal is to promote diversity among offspring.

- <u>Mutation Operators</u>

Three mutation operators were chosen for evaluation: binary mutation, swap mutation, and inversion mutation. However, the original operators provided in the practical classes were not suitable for this context:

- Swap mutation and inversion mutation are based on the idea that changing the order of genes alters the chromosome, but for our timetable representation, the order of classes does not affect the solution.
- Binary mutation is used for binary representations.

Therefore, the existing operators were modified to fit our requirements. In the adapted binary mutation, a gene is randomly selected, and one of its attributes is altered. For swap mutation, two classes from the timetable are randomly chosen, and half of their attributes are exchanged. Finally, inversion mutation involves randomly generating a class index and using this index to swap the original gene with a newly created random class.

Evaluation process

The genetic algorithm configurations were assessed across multiple runs and generations, considering different probabilities and population sizes. The goal was to evaluate all possible combinations, save the mean ABF (Average Best Fitness) for each, and ultimately compare them to identify the optimal combination. If a new combination exhibited a lower mean ABF than the current best, it would update the best combination and its corresponding ABF.

Results were meticulously saved in CSV files within designated folders. Detailed information on each parameter combination was recorded in Excel files, and comparison plots for the evaluated combinations were generated.

Data

Two datasets were prepared for testing: one simplistic and another more complex, to assess the algorithm's performance across varying levels of complexity. This approach was adopted to gauge the robustness of the proposed solution.

Tests and Results

Multiple parameter sets were created for test and improve the different combinations. All these parameters were tested on both simple and advanced datasets across all three fitness functions.

Parameters with no elitism: These initial tests aimed to understand the impact of excluding elitism. The results demonstrated poor algorithm performance without elitism, leading to the inclusion of elitism in subsequent tests. 24 combinations were tested with crossover probability equal to 0.9, mutation probability equal to 0.05 population size equal to 100 and 30 runs. The results are shown in *Figure 1* in annexes.

Base parameters with elitism: The comparison plots demonstrated that the combination of tournament selection, uniform crossover, and binary mutation was the most effective. As part of our analysis process, we made the decision to streamline our testing approach by focusing on the most promising combinations. After observing consistently subpar results and considering the computational overhead, we opted to remove abc and cycle crossover from further testing, resulting in 12 combinations, tested with crossover probability equal to 0.9, mutation probability equal to 0.05 population size equal to 100 and 30 runs. Incorporate elitism improved the overall convergence of the algorithm by preventing the loss of promising solutions. The results are shown in *Figures 2 to 7* in annexes.

After identifying the best combination, we encountered a greater challenge in achieving a fitness value of 0 when using the advanced dataset, and so we proceeded to try to optimize the results.

Population improvement: Increasing the population size to 200 slightly improved the results, but it was insufficient on its own.

Probability improvement: Adjusting the crossover and mutation probabilities to 0.8 and 0.2, respectively, pushing the values towards the upper and lower limits commonly used, did not yield better results, and was therefore discarded.

Comprehensive improvement: A comprehensive improvement strategy, involving a combination of increased population size, adjusted probabilities, and more runs, failed to yield significant improvements and proved to be computationally expensive.

Based on these evaluations, the best combination of parameters for was identified. The final set of parameters included tournament selection, uniform crossover, binary mutation, crossover probability equal to 0.9, mutation probability equal to 0.05, elitism, and a population size of 200.

The detailed fitness functions notably impact the convergence of the genetic algorithm. It's evident that as the complexity of the fitness function increases, the number of generations needed to reach optimal fitness values also increases, even when other factors remain constant. This is demonstrated by the gradual decline in the slopes of the curves.

Best

We conducted performance validation of the algorithm using exclusively the best parameters to ascertain their effectiveness. Additionally, we generated a visual representation of the final timetable, showcasing the optimal schedule produced by the algorithm. This visual aid served to validate the algorithm's efficacy and ensure that the generated schedules were both practical and conflict-free. The results are shown in *Figures 8 to 19* in annexes.

Conclusion

In general, we are satisfied with the achieved outcomes, having successfully identified optimal selection-crossover-mutation combinations that yielded a fitness function of 0 in most cases. One area we would like to further improve upon is addressing the empty spaces between classes, which we consider as a potential new constraint. Unfortunately, given the limited time we couldn’t implement this improvement.

Annexes

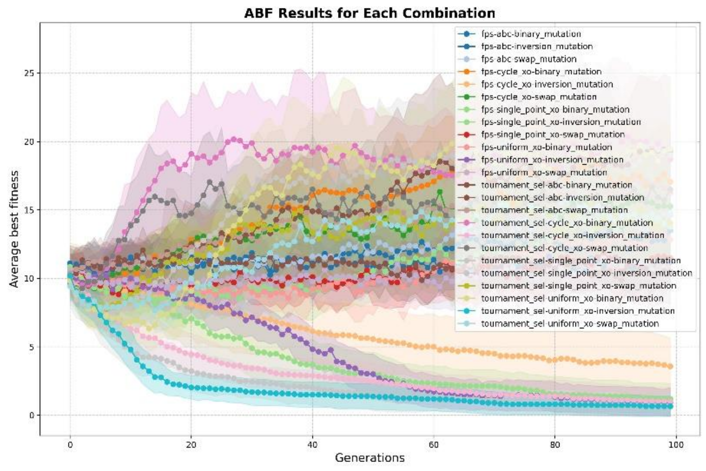


Fig.1 - Comparison plot for parameters with no elitism

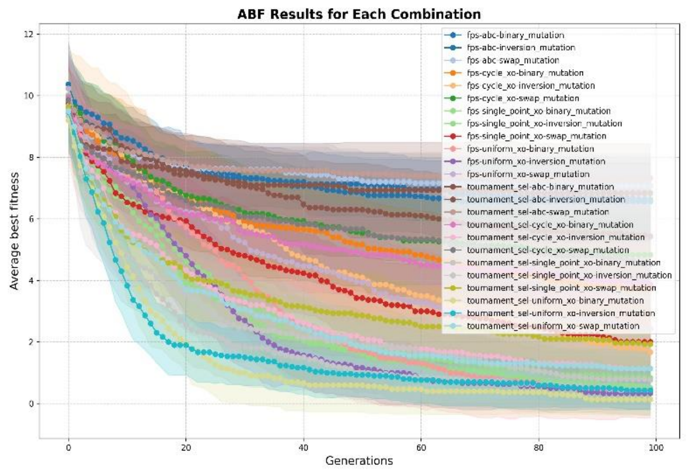


Fig. 2 - Comparison plot for base parameters, simple data and fitness 1

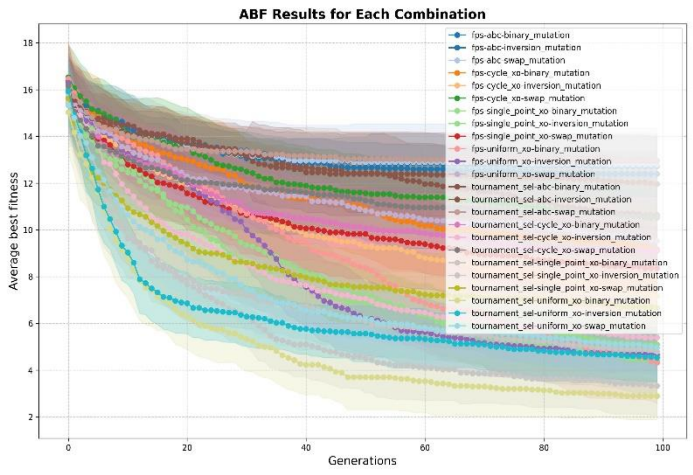


Fig. 3 - Comparison plot for base parameters, simple data and fitness 2

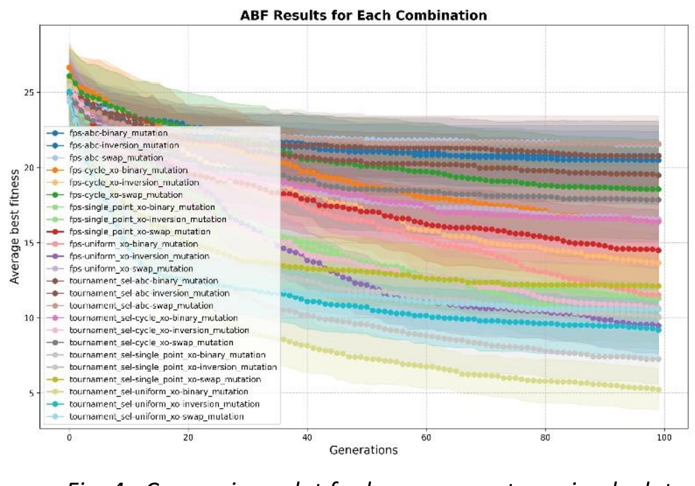


Fig. 4 - Comparison plot for base parameters, simple data and fitness 3

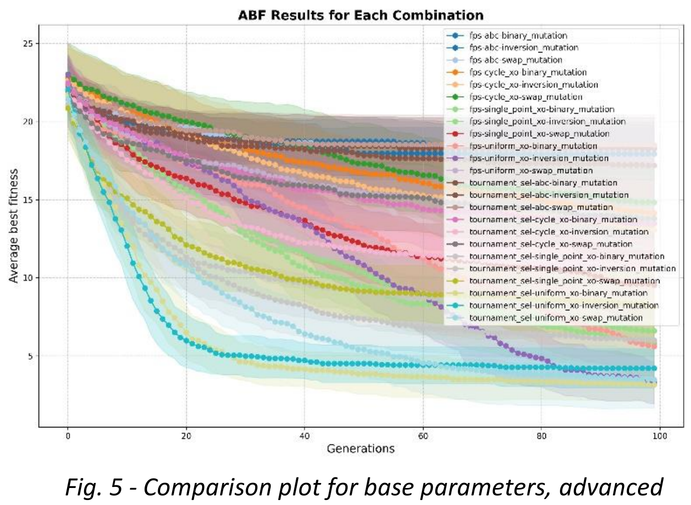


Fig. 5 - Comparison plot for base parameters, advanced data and fitness 1

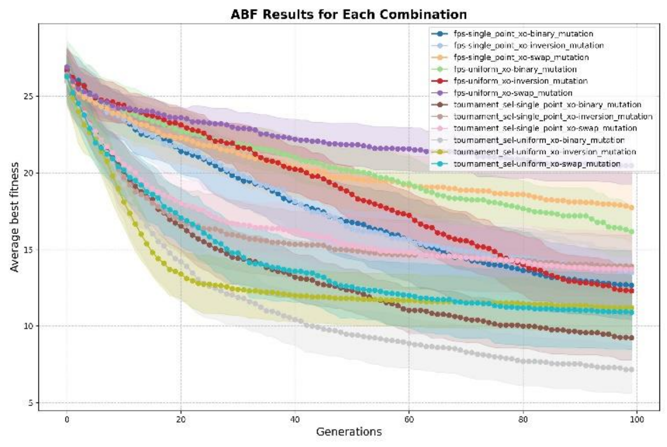


Fig. 6 - Comparison plot for base parameters, advanced data and fitness 2

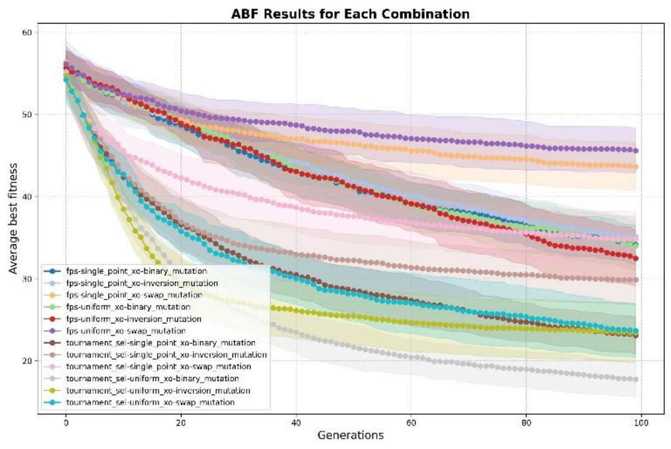


Fig. 7 - Comparison plot for base parameters, advanced data and fitness 3

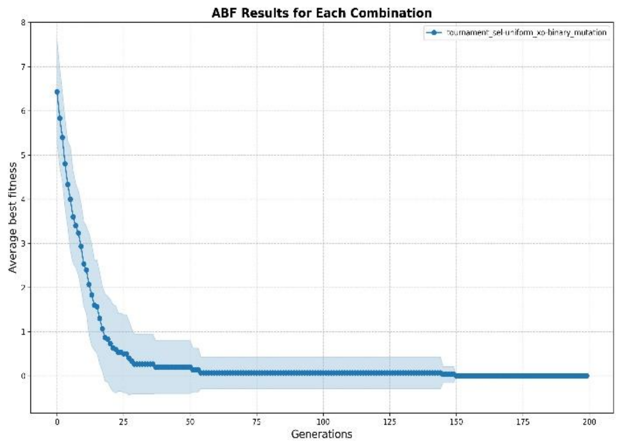
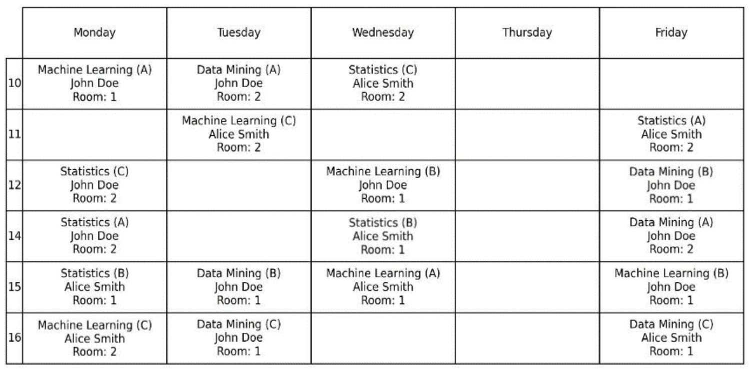


Fig. 8 - Plot for best parameters, simple data and fitness 1



	Monday	Tuesday	Wednesday	Thursday	Friday
10	Machine Learning (A) John Doe Room: 1	Data Mining (A) John Doe Room: 2	Statistics (C) Alice Smith Room: 2		
11		Machine Learning (C) Alice Smith Room: 2			Statistics (A) Alice Smith Room: 2
12	Statistics (C) John Doe Room: 2		Machine Learning (B) John Doe Room: 1		Data Mining (B) John Doe Room: 1
14	Statistics (A) John Doe Room: 2		Statistics (B) Alice Smith Room: 1		Data Mining (A) John Doe Room: 2
15	Statistics (B) Alice Smith Room: 1	Data Mining (B) John Doe Room: 1	Machine Learning (A) Alice Smith Room: 1		Machine Learning (B) John Doe Room: 1
16	Machine Learning (C) Alice Smith Room: 2	Data Mining (C) John Doe Room: 1			Data Mining (C) Alice Smith Room: 1

Fig. 9 – Final timetable generated using best parameters, simple data and fitness 1

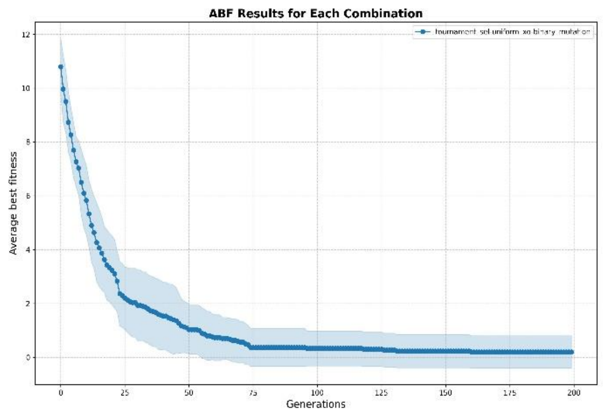
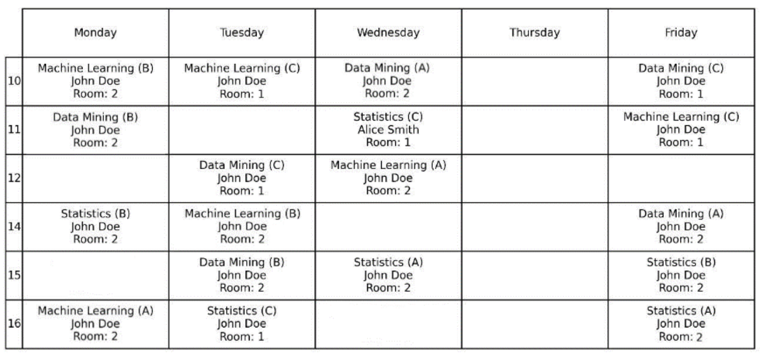


Fig. 10 - Plot for best parameters, simple data and fitness 2



	Monday	Tuesday	Wednesday	Thursday	Friday
10	Machine Learning (B) John Doe Room: 2	Machine Learning (C) John Doe Room: 1	Data Mining (A) John Doe Room: 2		Data Mining (C) John Doe Room: 1
11	Data Mining (B) John Doe Room: 2		Statistics (C) Alice Smith Room: 1		Machine Learning (C) John Doe Room: 1
12		Data Mining (C) John Doe Room: 1	Machine Learning (A) John Doe Room: 2		
14	Statistics (B) John Doe Room: 2	Machine Learning (B) John Doe Room: 2			Data Mining (A) John Doe Room: 2
15		Data Mining (B) John Doe Room: 2	Statistics (A) John Doe Room: 2		Statistics (B) John Doe Room: 2
16	Machine Learning (A) John Doe Room: 2	Statistics (C) John Doe Room: 1			Statistics (A) John Doe Room: 2

Fig. 11 – Final timetable generated using best parameters, simple data and fitness 1

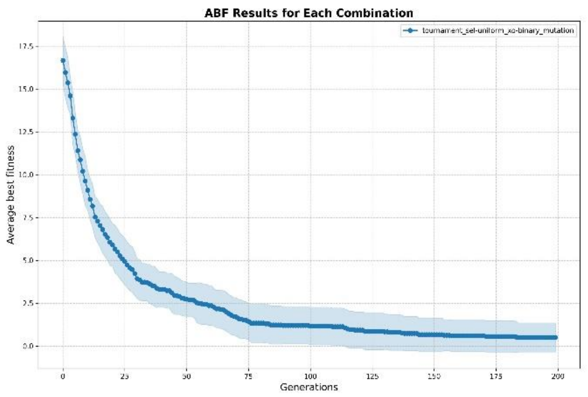
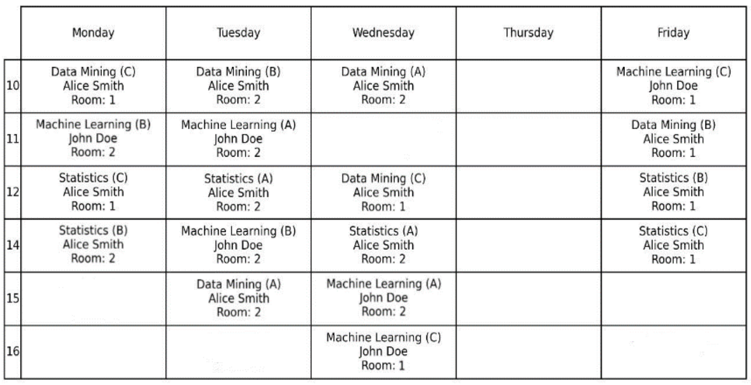


Fig. 12 - Plot for best parameters, simple data and fitness 3



	Monday	Tuesday	Wednesday	Thursday	Friday
10	Data Mining (C) Alice Smith Room: 1	Data Mining (B) Alice Smith Room: 2	Data Mining (A) Alice Smith Room: 2		Machine Learning (C) John Doe Room: 1
11	Machine Learning (B) John Doe Room: 2	Machine Learning (A) John Doe Room: 2			Data Mining (B) Alice Smith Room: 1
12	Statistics (C) Alice Smith Room: 1	Statistics (A) Alice Smith Room: 2	Data Mining (C) Alice Smith Room: 1		Statistics (B) Alice Smith Room: 1
14	Statistics (B) Alice Smith Room: 2	Machine Learning (B) John Doe Room: 2	Statistics (A) Alice Smith Room: 2		Statistics (C) Alice Smith Room: 1
15		Data Mining (A) Alice Smith Room: 2	Machine Learning (A) John Doe Room: 2		
16			Machine Learning (C) John Doe Room: 1		

Fig. 13 – Final timetable generated using best parameters, simple data and fitness 3

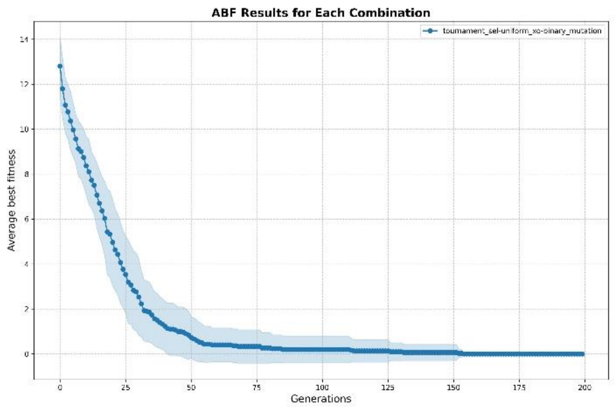
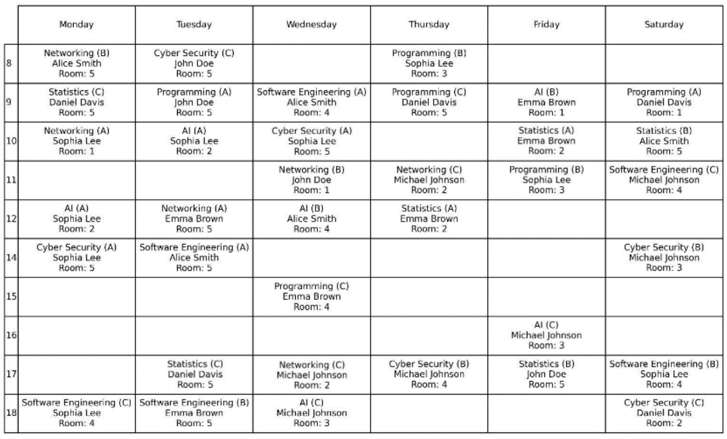


Fig. 14 - Plot for best parameters, advanced data and fitness 1



	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
8	Networking (B) Alice Smith Room: 5	Cyber Security (C) John Doe Room: 5		Programming (B) Sophia Lee Room: 3		
9	Statistics (C) Daniel Davis Room: 5	Programming (A) John Doe Room: 5	Software Engineering (A) Alice Smith Room: 4	Programming (C) Daniel Davis Room: 5	AI (B) Emma Brown Room: 1	Programming (A) Daniel Davis Room: 1
10	Networking (A) Sophia Lee Room: 1	AI (A) Sophia Lee Room: 2	Cyber Security (A) Sophia Lee Room: 5		Statistics (A) Emma Brown Room: 2	Statistics (B) Alice Smith Room: 5
11			Networking (B) John Doe Room: 1	Networking (C) Michael Johnson Room: 2	Programming (B) Sophia Lee Room: 3	Software Engineering (C) Michael Johnson Room: 4
12	AI (A) Sophia Lee Room: 2	Networking (A) Emma Brown Room: 5	AI (B) Alice Smith Room: 4	Statistics (A) Emma Brown Room: 2		
14	Cyber Security (A) Sophia Lee Room: 5	Software Engineering (A) Alice Smith Room: 5				Cyber Security (B) Michael Johnson Room: 3
15			Programming (C) Emma Brown Room: 4			
16					AI (C) Michael Johnson Room: 3	
17		Statistics (C) Daniel Davis Room: 5	Networking (C) Michael Johnson Room: 2	Cyber Security (B) Michael Johnson Room: 4	Statistics (B) John Doe Room: 5	Software Engineering (B) Sophia Lee Room: 4
18	Software Engineering (C) Sophia Lee Room: 4	Software Engineering (B) Emma Brown Room: 5	AI (C) Michael Johnson Room: 3			Cyber Security (C) Daniel Davis Room: 2

Fig. 15 – Final timetable generated using best parameters, advanced data and fitness 1

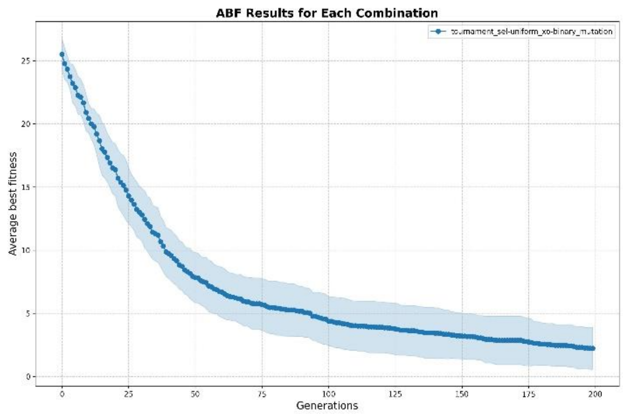
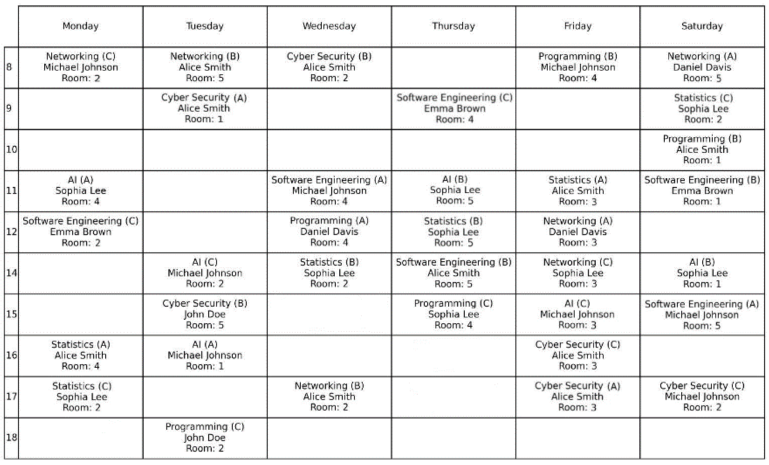


Fig. 16 - Plot for best parameters, advanced data and fitness 2



	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
8	Networking (C) Michael Johnson Room: 2	Networking (B) Alice Smith Room: 5	Cyber Security (B) Alice Smith Room: 2		Programming (B) Michael Johnson Room: 4	Networking (A) Daniel Davis Room: 5
9		Cyber Security (A) Alice Smith Room: 1		Software Engineering (C) Emma Brown Room: 4		Statistics (C) Sophia Lee Room: 2
10						Programming (B) Alice Smith Room: 1
11	AI (A) Sophia Lee Room: 4		Software Engineering (A) Michael Johnson Room: 4	AI (B) Sophia Lee Room: 5	Statistics (A) Alice Smith Room: 3	Software Engineering (B) Emma Brown Room: 1
12	Software Engineering (C) Emma Brown Room: 2		Programming (A) Daniel Davis Room: 4	Statistics (B) Sophia Lee Room: 5	Networking (A) Daniel Davis Room: 3	
14		AI (C) Michael Johnson Room: 2	Statistics (B) Sophia Lee Room: 2	Software Engineering (B) Alice Smith Room: 5	Networking (C) Sophia Lee Room: 3	AI (B) Sophia Lee Room: 1
15		Cyber Security (B) John Doe Room: 5		Programming (C) Sophia Lee Room: 4	AI (C) Michael Johnson Room: 3	Software Engineering (A) Michael Johnson Room: 5
16	Statistics (A) Alice Smith Room: 4	AI (A) Michael Johnson Room: 1			Cyber Security (C) Alice Smith Room: 3	
17	Statistics (C) Sophia Lee Room: 2		Networking (B) Alice Smith Room: 2		Cyber Security (A) Alice Smith Room: 3	Cyber Security (C) Michael Johnson Room: 2
18		Programming (C) John Doe Room: 2				

Fig. 17 – Final timetable generated using best parameters, advanced data and fitness 2

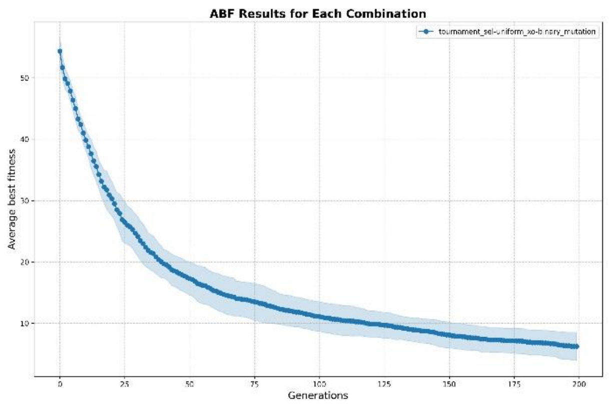
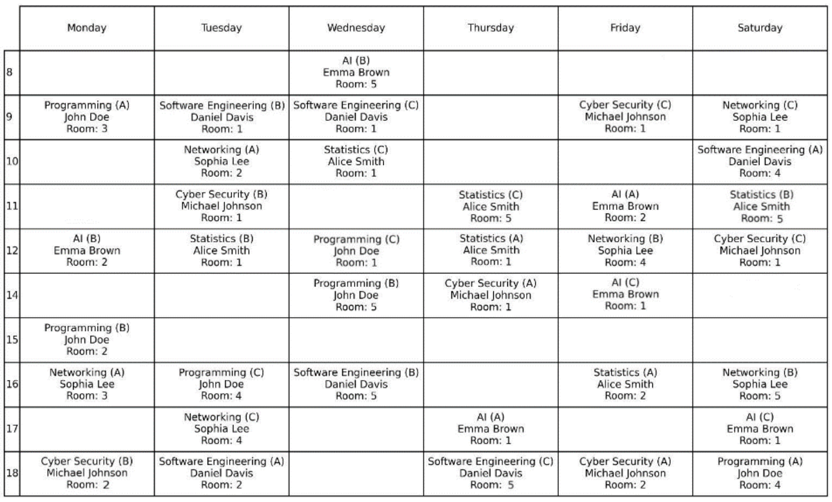


Fig. 18 - Plot for best parameters, advanced data and fitness 3



	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
8			AI (B) Emma Brown Room: 5			
9	Programming (A) John Doe Room: 3	Software Engineering (B) Daniel Davis Room: 1	Software Engineering (C) Daniel Davis Room: 1		Cyber Security (C) Michael Johnson Room: 1	Networking (C) Sophia Lee Room: 1
10		Networking (A) Sophia Lee Room: 2	Statistics (C) Alice Smith Room: 1			Software Engineering (A) Daniel Davis Room: 4
11		Cyber Security (B) Michael Johnson Room: 1		Statistics (C) Alice Smith Room: 5	AI (A) Emma Brown Room: 2	Statistics (B) Alice Smith Room: 5
12	AI (B) Emma Brown Room: 2	Statistics (B) Alice Smith Room: 1	Programming (C) John Doe Room: 1	Statistics (A) Alice Smith Room: 1	Networking (B) Sophia Lee Room: 4	Cyber Security (C) Michael Johnson Room: 1
14			Programming (B) John Doe Room: 5	Cyber Security (A) Michael Johnson Room: 1	AI (C) Emma Brown Room: 1	
15	Programming (B) John Doe Room: 2					
16	Networking (A) Sophia Lee Room: 3	Programming (C) John Doe Room: 4	Software Engineering (B) Daniel Davis Room: 5		Statistics (A) Alice Smith Room: 2	Networking (B) Sophia Lee Room: 5
17		Networking (C) Sophia Lee Room: 4		AI (A) Emma Brown Room: 1		AI (C) Emma Brown Room: 1
18	Cyber Security (B) Michael Johnson Room: 2	Software Engineering (A) Daniel Davis Room: 2		Software Engineering (C) Daniel Davis Room: 5	Cyber Security (A) Michael Johnson Room: 2	Programming (A) John Doe Room: 4

Fig. 19 – Final timetable generated using best parameters, advanced data and fitness 3