11753 - Computational Intelligence
Intelligent Systems
Universitat de les Illes Balears

COURSE-PROFESSOR-CLASS ASSIGNMENTS WITH EVOLUTIONARY ALGORITHM

Manasut, Phornphawit
Charnota, Sofiia
Tomeu

Supervisor:
Massanet, Sebastián

1st Semester

January 13, 2024

Contents

1 Introduction

1.1 Problem Statement

The central focus of this assignment is the implementation of an evolutionary algorithm devised to address the intricate task of schedule creation within a high school environment. The objective is to develop an efficient and optimal algorithm capable of generating weekly schedules for each class, considering a multitude of constraints and requirements inherent to educational institutions.

1.2 Assumptions

A few assumptions lay the groundwork for this assignment. They are:

1. All classes operate within a uniform study period from Monday to Friday, spanning 8 AM to 5 PM. Basically school period starts from 8AM and last course should start at 4 PM.
2. The total study hours per week of a class should match up to total hours required from all courses for that class. X
3. Each session of a course lasts 1 hour.

1.3 Hard Constraints

The algorithm operates under a set of unyielding hard constraints to ensure the practicality and viability of the generated schedules. Two critical hard constraints define the limits of the algorithm:

1. **Professor Conflicts:** Professors are prohibited from concurrently teaching two classes.
2. **Class Conflicts:** Each class must not host more than one course at any given time.

1.4 Soft Constraints

To enhance the adaptability and fairness of the algorithm, a set of soft constraints has been introduced:

1. **Satisfy Course hours:** The evaluation function aim to satisfy course hours of each classes by giving extra fitness to those with satisfied course hours.
2. **Professor Equity:** Emphasizing a comparable number of total teaching hours among professors, this constraint aims to ensure an equitable distribution of teaching responsibilities among the faculty.
3. **Satisfy Course hours:** The evaluation function aim to satisfy course hours of each classes by giving extra fitness to those with satisfied course hours.
4. **No large gaps:** The studying period for each class should not have too many gaps such that the student loses focus from their studies.

5. **Breaks:** For both students and lecturers, they should not be studying or teaching for too many hours. In our case, we define too many hours as anything more than 3 consecutive hours.

6. **Earlier Weekends:** It is good for students to start their weekends earlier as our scheduling algorithm give bonus to class schedules that end earlier in the week.

2 Implementation

2.1 Data Representation

The data representation is the most important part of evolutionary algorithm. Without proper data representation, experimenting new selection, crossover, mutations, and fitness strategies can be difficult. This is especially true for the given task as it is very complicated. Hence, object-oriented style is employed for the given task.

2.1.1 Time Slot

Time slot is represented as a tuple of `(day, time)` where both variables `day` and `time` are integer. In our experiment, we represent day as 0 to 6 (inclusive) which corresponds to Monday till Sunday and time is presented as 0 to 8 which corresponds to 8 a.m. till 4 p.m., there are no restrictions and user can change this as they wish. However, this may have impact on performances as the current strategy employed favors courses that are assigned earlier in the week. Therefore, users should keep this in mind that time slot representation can have impact on model performance. This representation of time slot is good in that we can turn these 2 integers into a single integer and calculate the first and last course of the week for a class. As such, the fitness function can reward different kind of course timing.

2.1.2 Professor

`Professor` class contains following attributes: `(professor_id, available_slots)`. `professor_id` is unique identifier for the professor. `available_slots` are list of time slots that the professor is available to teach.

2.1.3 Course

`Course` class contains following attributes: `(course_id, professors)`. `course_id` is unique identifier for the professor. `professors` is a list of `Professor` objects that contains the professors that are available to teach this course.

2.1.4 Class

`Class` class contains following attributes: `(class_id, total_weekly_hours, courses, time_slots)`. `class_id` is unique identifier for the professor. `time_slots` is a list of time slot that the class is available. `total_weekly_hours` is total number of

2.1.5 Entry

An entry is a tuple in the form of `(time slot, class id, course id, professor id)`. In layman term, an entry tells the user that this class is supposed to be having a course with this professor at this time slot. Entry is not represented as a class in our implementation but it is worth explaining be because it is an integral part of the schedule.

2.1.6 Schedule

A `Schedule` class represents an individual in population in evolutionary algorithm term. Its attributes include a list of entries, a dictionary mapping class id to the class object, a dictionary mapping professor id to professor object, and a dictionary mapping the courses to course object. The ever-changing attribute is the list of entries while the rest of attributes will face no change after the schedule initialization since the current availability of the courses, classes, and professors in a schedule can be inferred from the list of entries.

A Schedule object has following functions:

1. add_entry: Add a new entry to the list of entries.
2. is_class_course_hour_satisfied: Return a Boolean stating if all classes' course hours requirement is satisfied given current entries.
3. is_class_total_hour_satisfied: Return a Boolean stating if all classes' total hours requirement is satisfied given current entries.
4. difference_class_course_hour: Return the sum the absolute of differences between current assigned course hour of each class and the expected assigned course hour.
5. difference_class_total_hour: Sum the absolute of differences between each class total hours and the expected total hours.
6. is_hard_constraints_satisfied: Return a Boolean stating if all hard constraints are satisfied given current entries.
7. correct_schedule: Perform the function remove_conflicting_entries and rescheduling_entries
8. remove_conflicting_entries: Remove conflicting entries from the list of entries in current schedule.
9. reschedule_entries
   (a) Create new entries after taking current entries into consideration.
   (b) This is mostly used after removing conflicting entries in order to satisfy the course requirements.
   (c) The strategy is to generate tuples of course id and class id with repetitions up to the number of hours left required for the class to take that course. Then shuffle these tuple before iterating them to randomly assign available professors to teach the class. Hard constraints are considered in this process.
   (d) It is possible that rescheduling of entries are not possible to satisfy all the courses required for the class to take and therefore, stale loop limit is implemented.

(e) It is possible that total course time may not be met but that's a constraint that we are willing to overlook and that is why it is a soft constraint.

(f) This above strategy is very similar to the generation of new random schedules.

Function 1 is created to follow OOP convention. Functions 2 to 5 are used for fitness evaluation. Functions 6 to 9 are used to make sure hard constraints are adhered to.

2.1.7 Justification for Schedule Representation

The main reason to keep a list of entries as a schedule representation is simple. It is to keep things simple and save debugging time. With functions like crossover and mutations, it is very unrealistic to keep changing the different other objects and the code can get convoluted. Since we can derive current availability of other attributes with thge list of entries, there is no need to make things complicated. It is a rule in software engineering to keep things simple and as such, this schema is chosen. Below are scenarios where this representation is useful.

1. Crossover function only requires swapping entries of 2 parents to result in a child is just as simple as list slicing. Afterwards, schedule correction function is applied to ensure that the hard constraints are kept. With corrections being random assignments, this may allow us to perform some mutations on the children innately and jump out of local maxima.

2. Mutation function only needs to pick a single entry to change and corrections to the schedule can be made afterwards.

2.2 Crossover Function

In this implementation, the crossover starts by converting the schedule attributes (classes, professors, courses) from dictionaries to lists. This step simplifies the subsequent merging process, as working with lists is more straightforward than with dictionaries. These lists are used to initialize a new Schedule object, which serves as the child schedule. This approach ensures that the child inherits the structural framework of its parents, including all available classes, professors, and courses.

The core of the crossover process involves combining the schedule entries from both parents. The entries from the first half of the first parent's schedule are concatenated with the entries from the second half of the second parent's schedule. This method creates a child schedule that inherits characteristics from both parents. The choice of a half-point for merging is a common practice in genetic algorithms, as it provides a balanced mix of both parents' traits.

Finally, the newly formed child schedule is passed through a correction process to ensure it adheres to hard constraints, such as preventing a class or professor from being double-booked. This step is crucial because the straightforward merging of two parent schedules might result in a child schedule that violates these constraints. The `correct_schedule` method adjusts the child schedule to maintain its feasibility within the defined rules of the scheduling problem. The result is a new, viable schedule that combines aspects of both its parents while respecting the essential constraints of the scheduling environment.

As stated earlier and demonstrated here, having a list of entries as our representation of schedule simplifies the crossover function.

2.3 Mutation Function

The `mutate_schedule` function in evolutionary algorithms introduces variability into a schedule by randomly removing an entry and then adjusting the remaining schedule to maintain its validity. This process starts by selecting and removing a random entry (a scheduled class, course, or professor) from the schedule. After this removal, which creates a gap in the schedule, the function then invokes a correction method to realign the remaining entries while ensuring adherence to the hard constraints of the scheduling problem, such as avoiding double bookings. This mutation mechanism is crucial for introducing diversity into the population of schedules, helping the algorithm explore a wider range of potential solutions and avoid being trapped in local optima.

As stated earlier and demonstrated here, having a list of entries as our representation of schedule simplifies the mutation function.

2.4 Selection Strategy

Initially, we implemented 2 different selection strategies but they have been observed in our experiments to have impact on arriving at the better solutions at the same number of generations. Therefore, we implemented 4 different selection strategies to use in our scenarios to see which one of them are the best for each scenarios given the same number of parameters. For all strategies, given a list of fitness values, minimum fitness values are added to each fitness value before the random selection is applied. This is to avoid negative fitness value when assigning weights to schedules for random choosing.

2.4.1 Selection Strategy: Random Weighted

In this strategy, each schedule, s, is randomly selected with probability p_s where p_s is proportion of fitness of schedule s as compared to total fitness summed up from all schedules.

2.4.2 Selection Strategy: Battle Royale Tournament

In this strategy, the number of participants, p, is predefined. p number of schedules are selected randomly and then random selection is applied among these individuals with weighted fitness to choose the winner as a parents.

2.4.3 Selection Strategy: Single Elimination Tournament

The SingleEliminationTournamentSelection strategy is used in evolutionary algorithms to select a subset of promising candidates (or parents) from a larger population for further breeding. It mimics the structure of a single-elimination tournament, often seen in sports competitions, where participants are eliminated after each round until a final winner is determined.

When this selection strategy is invoked, it first randomly selects a specified number of participants `n_participants` from the population. These selected individuals are then paired up to compete against each other. The winner of each pair is chosen based on their fitness levels, with higher fitness increasing the likelihood of being selected. This process of pairing and selecting winners continues through multiple rounds until only one individual remains.

The final winner of the tournament is then chosen as a parent. This process is repeated until the desired number of parents `num_parents` is selected. This strategy effectively balances the exploration of the population with the exploitation of fitter individuals, ensuring that individuals with higher fitness have a higher chance of being selected for breeding while maintaining diversity in the selection process.

2.4.4 Selection Strategy: Round Robin Tournament

The RoundRobinTournamentSelection class in evolutionary algorithms implements a selection strategy modeled after a round-robin tournament, often used in sports and games. This approach involves each participant competing against every other participant, ensuring a comprehensive assessment of the pool.

Upon invoking this strategy, it randomly selects a fixed number `n_participants` from the population for inclusion in the tournament. Each selected individual then competes against every other participant in turn. In each matchup, a winner is determined through a random choice weighted by the fitness of the competitors, simulating a competitive scenario where individuals with higher fitness are more likely to win.

After all matchups are completed, the individual with the highest number of wins (the highest score) is selected as a parent. This process repeats until the desired number of parents `num_parents` is selected. The Round Robin selection method is notable for its thoroughness in evaluating individuals, as every participant competes against all others, allowing for a well-rounded selection based on overall performance.

2.5 Evaluation Function

Initially, the function assigns a high base score if the schedule satisfies two fundamental requirements: matching the total weekly hours for each class and fulfilling the weekly hour requirements for each course within those classes. This base score acts as a foundational assessment, ensuring that the schedule meets the basic educational needs.

The function then adds bonuses for schedules that allow classes to finish earlier in the week. It calculates the difference between the latest time slot in each class's schedule and the maximum available time slot for that class. Schedules enabling classes to finish sooner receive higher bonuses, promoting more concentrated and potentially effective learning periods. Letting students and professors end their study/work week earlier should allows them to have more leisure time.

On the flip side, the function imposes penalties for undesirable scheduling patterns. One such penalty is for large gaps between classes, which can disrupt the learning process and reduce efficiency. This penalty is calculated by a separate function, `penalize_large_gap`, which identifies significant breaks in the daily class schedule.

Further penalties are applied for schedules that overburden professors or classes with consecutive hours without breaks. Excessive consecutive teaching hours for professors and consecutive class hours without breaks are penalized, as these conditions can lead to diminished teaching effectiveness and student fatigue.

An additional penalty is based on the standard deviation of teaching hours across all professors. This penalty aims to ensure a fair distribution of teaching workload among faculty members. A higher standard deviation, indicating a more uneven distribution of teaching hours, results in a greater penalty.

Finally, the function imposes penalties for any discrepancies between the scheduled hours and the required hours for both individual courses and the total class hours. These penalties ensure that the schedule not only meets the basic requirements but also adheres closely to the ideal distribution of teaching hours across various courses and classes. This comprehensive evaluation approach ensures that the fitness score reflects both the effectiveness and the practicality of the schedule.

This process of penalizing and rewards should be addressing all the soft constraints stated in the earlier section.

3 Experiment Setups

In this chapter, we delve into the detailed experimental setups designed to thoroughly evaluate the evolutionary algorithm's performance in generating schedules for a high school environment. The chapter is divided into two distinct scenarios, each crafted to assess the algorithm under different levels of complexity and constraints.

3.1 Scenario 1

Scenario 1 serves as the foundational experiment, providing insights into the algorithm's performance under easy conditions. They are shown in Table 1

Class ID	Total Weekly Hours	Courses
1	5	Math (3 hours), Science (2 hours)
2	5	Math (2 hours), Science (3 hours)

Table 1: Classes in Scenario 1

Two professors are introduced in this scenario, each with specific time availabilities.

1. **Professor Juan:** Available Monday to Friday. Available Hours: 8 AM to 11 AM.
2. **Professor Carlos:** Available Monday to Friday. Available Hours: 11 AM to 4 PM.

The time slots represent the discrete units of time available during a typical school week. The defined time slots cover all weekdays (Monday to Friday) and range from 8 AM to 4 PM.

Two courses are offered in this scenario:

1. **Math:** Taught by Professor Juan and Professor Carlos. Weekly Hours: 3 for Class 1, 2 for Class 2.
2. **Science:** Taught by Professor Carlos. Weekly Hours: 2 for Class 1, 3 for Class 2.

3.2 Scenario 2

This scenario is a slightly more extensive scenario compared to scenario 1 to test robustness of the algorithm.

Class ID	Total Weekly Hours	Courses
1	5	Math (3 hours), Science (2 hours)
2	5	Math (2 hours), Science (3 hours)
4	4	History (2 hours), Art (2 hours)
5	6	History (3 hours), Art (3 hours)

Table 2: Classes in Scenario 2

The professorial lineup is an extension of Scenario 1, encompassing familiar faces and introducing new educators with distinct availability patterns.

1. **Professor Juan:** Available Monday to Friday. Available Hours: 8 AM to 11 AM.
2. **Professor Carlos:** Available Monday to Friday. Available Hours: 11 AM to 4 PM.
3. **Professor Maria**: Available Monday to Friday, 1 PM to 4 PM.
4. **Professor Jose**: Available Monday to Friday, 9 AM to 12 PM.

The time slots structure remains consistent with Scenario 1, encompassing all weekdays and hours from 8 AM to 4 PM. This allows for comprehensive coverage and scheduling flexibility.

Courses include carryovers from Scenario 1, such as Math and Science, while introducing new subjects like History and Art.

1. **Math** (Inherited from Scenario 1): Taught by Professor Juan and Professor Carlos
2. **Science** (Inherited from Scenario 1): Taught by Professor Carlos
3. **History:** Taught by Professor Maria and Professor Jose **Art:** Taught by Professor Jose

3.3 Scenario 3

This scenario try to mimic real world scheduling problem. Each class have their own combinations of courses to take. Their total hours of courses are similar.

Class ID	Total Weekly Hours	Courses
1	18	Math (4 hours), Science (4 hours) History (4 hours), English (4 hours), PE (2 hours)
2	18	Math (4 hours), Science (4 hours) Art (4 hours), English (4 hours), PE (2 hours)
3	18	Math (4 hours), History (4 hours) Art (4 hours), English (4 hours), PE (2 hours)
4	18	Science (4 hours), History (4 hours) Art (4 hours), English (4 hours), PE (2 hours)

Table 3: Classes in Scenario 3

Here is the list of professors for this scenario.

1. **Professor Juan:** Available Monday to Friday. Available Hours: 8 AM to 11 AM. Also available from 1 PM to 4 PM on Monday and Tuesday.
2. **Professor Carlos:** Available Monday to Friday. Available Hours: 11 AM to 4 PM.
3. **Professor Maria**: Available Monday to Friday, 1 PM to 4 PM.
4. **Professor Wit**: Available Monday to Wednesday, 8 AM to 4 PM.
5. **Professor Sia**: Available Monday to Friday, 9 AM to 1 PM.
6. **Professor Mina**: Available Monday to Thursday, 11 AM to 4 PM.
7. **Professor Lebron**: Available Monday to Friday, 8 AM to 14 PM.

Here is the list of courses and the professors teaching them.

1. **Math** : Taught by Professor Juan and Professor Carlos
2. **Science**: Taught by Professor Juan Professor Carlos
3. **History:** Taught by Professor Maria and Professor Wit
4. **Art:** Taught by Professor Mina
5. **English:** Taught by Professor Sia
6. **PE:** Taught by Professor Lebron

3.4 Statistical Analysis

In this context, a series of statistical tests are performed to analyze the effectiveness of different selection strategies used in an evolutionary algorithm for scheduling. The process begins with the execution of multiple simulation runs, each employing a distinct selection strategy under a given configuration. The results of these simulations, specifically the best fitness score produced in each run, are compiled for statistical analysis for each selection strategy.

The first step in the analysis is to visually inspect the fitness distributions of the different strategies with a stacked bar plot, providing an immediate visual comparison of how the fitness scores are distributed across different strategies. Such visualizations can offer insights into the variability and central tendencies of the fitness scores associated with each strategy.

Next, a normality test is performed. This test checks whether the fitness scores for each strategy follow a normal distribution, a common assumption in many statistical tests. This step is crucial because the choice of subsequent statistical tests depends on whether the data satisfies this assumption. Based on the outcome of the normality test, either an ANOVA (Analysis of Variance) or a Kruskal-Wallis test is conducted. ANOVA is used if the normality assumption holds for all strategies; otherwise, the Kruskal-Wallis test is applied. Both tests aim to determine if there are statistically significant differences in the fitness scores across different strategies. However, ANOVA assumes normally distributed data and homogeneity of variances, while the Kruskal-Wallis test does not and is thus used as a non-parametric alternative.

Finally, if significant differences are detected by the ANOVA or Kruskal-Wallis test, the statistical test is used to conduct pairwise comparisons between the strategies. This step identifies which specific pairs of strategies differ significantly in their fitness scores. The implementation adjusts its testing approach based on the normality of the data, using either t-tests (for normally distributed data) or Man-Whitneyu tests (for data not following a normal distribution).

Overall, this sequence of statistical tests provides a comprehensive evaluation of the comparative effectiveness of different selection strategies used in the simulations, from a general assessment of differences (ANOVA or Kruskal-Wallis) to detailed pairwise comparisons.

4 Results

For each scenario, 2 types of results are provided. Firstly, the best schedule per scenario per selection strategy is provided. This is in form a a table with columns of Monday to friday and rows indexed by hours and then class id. Second version of professor id indexed instead of class id is also given for professor to use. Since there are too many tables to put in this pdf, only round robin results are given in appendix section since the strategy performs the best in this experiment. Refer to the csvs provided for more results. The statistical results from the process described previously are provided in this result section.

The parameters changed between each scenario is the number of generations, with 100 for the first scenario, 200 for the second scenario and 300 for the last scenario. The tournament size is 16 for all scenarios for the sake for fairness. The increase in number of generations is due to increase in complexity of the problem.

4.1 Result from Scenario 1

As we can see from the tables in Appendix section that most of the best schedules generated aim to finish the class earlier while not double booking. However, there are some soft constraints that were not met like class 1 having too 5 consecutive classes in a row on Monday - please refer to csv file for the best result for scenario 1 with Single Elimination Strategy. However, this issue is not seen in the provided result from Round Robin Strategy in the appendix section Table 10.

The results from normality tests for each strategy found no significance and therefore, normality is assumed. Hence, ANOVA test is applied which returns p-value of 9.0543e-05 which is statistically significant and that there are differences between each strategy in term of best schedule generated. Therefore, pairwise tests are employed and found pairs with statistically significant results detailed in Table 5. Inferring from Figure 1, we can roughly see that round robin strategies outperform the other 3 strategies.

	p-value	follow_normality
Random Weighted	0.3744	True
Battle Royale	0.5676	True
Round Robin	0.0559	True
Single Elimiation	0.9286	True

Table 4: Statistical Results for normality checks.

Figure 1: Stacked bar plot of best-fitness distribution per strategy for scenario 1.

	strategy1	strategy2	is_different	p-value
0	Random Weighted	Battle Royale	False	0.2602
1	Random Weighted	Round Robin	True	0.0027
2	Random Weighted	Single Elimiation	False	0.0524
3	Battle Royale	Round Robin	True	0.0001
4	Battle Royale	Single Elimination	True	0.0020
5	Round Robin	Single Elimination	False	0.2025

Table 5: Statistical Results for pairwise test using t-test with p-value statistical threshold of 0.05.

4.2 Result from Scenario 2

With scenario 2, the search space to find optimal solution becomes much larger, the differences between results from each strategy can be clearly seen from Figure 2. Kruskal-Wallis is employed with resulting p-value of 6.29e-52 and therefore, Man-Whitney U test is used for pairwise-comparisons and the results can be observed in Table 7. Here, we can clearly see that Round Robin Strategy is the best selection method.

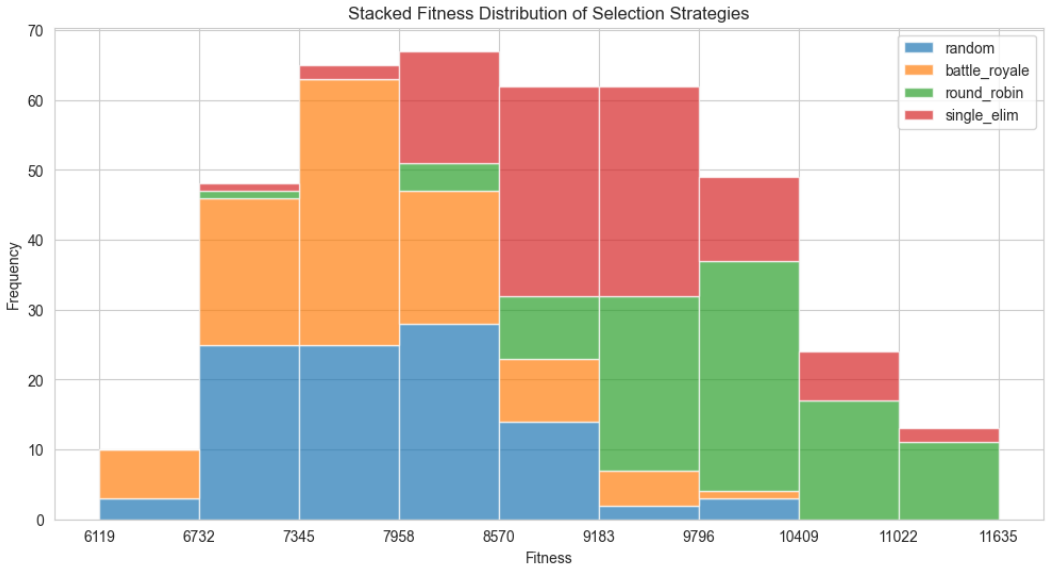

Figure 2: Stacked bar plot of best-fitness distribution per strategy for scenario 2.

	p-value	follow_normality
Random Weighted	0.0166	False
Battle Royale	0.1126	True
Round Robin	0.0450	False
Single Elimination	0.7908	True

Table 6: Statistical Results for normality checks for scenario 2.

	strategy1	strategy2	is_different	p-value
0	Random Weighted	Battle Royale	False	3.35e-01
1	Random Weighted	Round Robin	True	9.05e-30
2	Random Weighted	Single Elimination	True	7.70e-22
3	Battle Royale	Round Robin	True	2.71e-30
4	Battle Royale	Single Elimination	True	1.88e-23
5	Round Robin	Single Elimination	True	2.90e-10

Table 7: Statistical Results for pairwise test using t-test with p-value statistical threshold of 0.05 for scenario 2.

4.3 Result from Scenario 3

Since scenario 3 is quite complex and follow real world class scenario, the results from each strategy vastly differed from each other in term of best fitness as seen in Figure 3. With result from Table 8, we use ANOVA test and it returns p-value of 1.13e-45 and therefore, t-test is then used for pairwise comparisons resulting in Table 9. With Figure 3, we can safely infer that the Round Robin Selection strategy gives the us the best result. The distribution of Round Robin results are much more different than others as compared to when we were at scenario 1. This is due to the fact that this scenario is quite difficult with large search space.

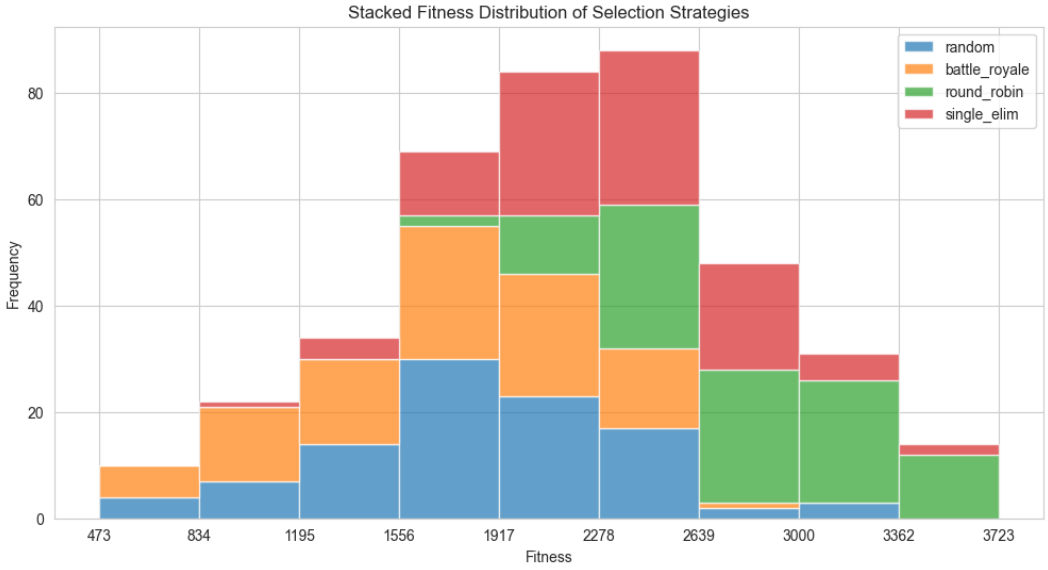

Figure 3: Stacked bar plot of best-fitness distribution per strategy for scenario 3.

	p-value	follow_normality
Random Weighted	0.5481	True
Battle Royale	0.2114	True
Round Robin	0.2611	True
Single Elimination	0.9955	True

Table 8: Statistical Results for normality checks for scenario 3.

	strategy1	strategy2	is_different	p-value
0	Random Weighted	Battle Royale	True	4.57e-02
1	Random Weighted	Round Robin	True	6.67e-28
2	Random Weighted	Single Elimination	True	7.05e-10
3	Battle Royale	Round Robin	True	3.25e-35
4	Battle Royale	Single Elimination	True	1.03e-15
5	Round Robin	Single Elimination	True	7.43e-11

Table 9: Statistical Results for pairwise test using t-test with p-value statistical threshold of 0.05 for scenario 3.

5 Conclusion

In conclusion, in my opinion, evolutionary algorithm is quite an effective method to solve this problem since soft constraints can be adjusted accordingly based on the fitness function. The penalty and bonus gives the algorithm quite an incentive to solve this issue. In addition, we experimented multiple selections strategies and managed to statistically conclude that the Round Robin Tournament strategy is the best at giving us the highest fitness schedule. In addition, the round robin selection strategy managed to create very satisfied schedule for each classes. If we see below in Appendix section, all classes from scenario 1 and 2 ended within 2 days of study and all classes in scenario 3 ended within 4 days. This is not the case if the reader checked out the results from other strategy provided in the various csvs.

However, there is a clear issue with our statistical analysis method. This is because we did not utilized statistical correction methods for our statistical tests. Despite this issue, since the result is consistent for all scenarios, it can be overlooked.

6 Appendix

Some tables are too long to be shown here, the readers may find better experience in referring to the csvs provided.

6.1 Result for Scenario 1

Hour	Class ID	Monday	Tuesday	Wednesday	Thursday	Friday
8 AM	1		(Math, Juan)			
	2	(Math, Juan)				
9 AM	1					
	2	(Math, Juan)				
10 AM	1	(Math, Juan)	(Math, Juan)			
	2					
11 AM	1	(Science, Carlos)	(Science, Carlos)			
	2					
12 PM	1					
	2	(Science, Carlos)				
1 PM	1					
	2	(Science, Carlos)				
2 PM	1					
	2	(Science, Carlos)				
3 PM	1					
	2					
4 PM	1					
	2					

Table 10: Best Schedule for Scenario 1 with round robin strategy with class indexed.

Hour	Professor ID	Monday	Tuesday	Wednesday	Thursday	Friday
8 AM	Juan	(Math, 2)	(Math, 1)			
	Carlos					
9 AM	Juan	(Math, 2)				
	Carlos					
10 AM	Juan	(Math, 1)	(Math, 1)			
	Carlos					
11 AM	Juan					
	Carlos	(Science, 1)	(Science, 1)			
12 PM	Juan					
	Carlos	(Science, 2)				
1 PM	Juan					
	Carlos	(Science, 2)				
2 PM	Juan					
	Carlos	(Science, 2)				
3 PM	Juan					
	Carlos					
4 PM	Juan					
	Carlos					

Table 11: Best Schedule for Scenario 1 with round robin strategy with professor indexed.

Hour	Class ID	Monday	Tuesday	Wednesday	Thursday	Friday
8 AM	1		(Math, Juan)			
	2	(Math, Juan)				
	3					
	4					
9 AM	1		(Math, Juan)			
	2	(Math, Juan)				
	3	(History, Jose)				
	4					
10 AM	1					
	2					
	3					
	4					
11 AM	1					
	2	(Science, Carlos)				
	3					
	4	(Art, Jose)	(Art, Jose)			
0 PM	1	(Science, Carlos)				
	2		(Science, Carlos)			
	3	(Art, Jose)				
	4		(Art, Jose)			
1 PM	1					
	2					
	3	(Art, Jose)				
	4					
2 PM	1	(Math, Carlos)				
	2		(Science, Carlos)			
	3	(History, Maria)				
	4		(History, Maria)			
3 PM	1	(Science, Carlos)				
	2					
	3					
	4	(History, Maria)	(History, Maria)			
4 PM	1					
	2					
	3					
	4					

Table 12: Best Schedule for Scenario 2 with round robin strategy with class indexed.

Hour	Professor ID	Monday	Tuesday	Wednesday	Thursday	Friday
8 AM	Juan	(Math, 2)	(Math, 1)			
	Carlos					
	Maria					
	Jose					
9 AM	Juan	(Math, 2)	(Math, 1)			
	Carlos					
	Maria					
	Jose	(History, 3)				
10 AM	Juan					
	Carlos					
	Maria					
	Jose					
11 AM	Juan					
	Carlos	(Science, 2)				
	Maria					
	Jose	(Art, 4)	(Art, 4)			
0 PM	Juan					
	Carlos	(Science, 1)	(Science, 2)			
	Maria					
	Jose	(Art, 3)	(Art, 4)			
1 PM	Juan					
	Carlos					
	Maria					
	Jose	(Art, 3)				
2 PM	Juan					
	Carlos	(Math, 1)	(Science, 2)			
	Maria	(History, 3)	(History, 4)			
	Jose					
3 PM	Juan					
	Carlos	(Science, 1)				
	Maria	(History, 4)	(History, 4)			
	Jose					
4 PM	Juan					
	Carlos					
	Maria					
	Jose					

Table 13: Best Schedule for Scenario 2 with round robin strategy with professor indexed.

Hour	Class ID	Monday	Tuesday	Wednesday	Thursday	Friday
8 AM	1		(Science, Juan)	(History, Wit)		
	2	(Science, Juan)		(PE, Lebron)	(PE, Lebron)	
	3	(PE, Lebron)		(Math, Juan)		
	4					
9 AM	1	(History, Wit)	(English, Sia)	(Science, Juan)		
	2	(English, Sia)	(Science, Juan)	(English, Sia)		
	3	(Math, Juan)			(English, Sia)	
	4		(PE, Lebron)			(English, Sia)
10 AM	1		(English, Sia)	(Math, Juan)		
	2		(Math, Juan)			
	3	(English, Sia)				
	4	(Science, Juan)	(History, Wit)			(PE, Lebron)
11 AM	1	(Math, Carlos)	(English, Sia)			
	2	(Art, Mina)	(Science, Carlos)	(Art, Mina)		
	3			(English, Sia)	(English, Sia)	
	4	(History, Wit)			(Art, Mina)	(English, Sia)
12 PM	1	(PE, Lebron)	(PE, Lebron)	(Math, Carlos)		
	2	(Science, Carlos)	(Art, Mina)	(English, Sia)		
	3	(Art, Mina)	(Math, Carlos)	(Art, Mina)	(Art, Mina)	
	4					(English, Sia)
1 PM	1	(English, Sia)	(History, Maria)	(Math, Carlos)		
	2	(Art, Mina)	(Math, Juan)	(English, Sia)		
	3	(Math, Carlos)	(Art, Mina)	(PE, Lebron)		
	4	(Science, Juan)	(English, Sia)	(Art, Mina)	(Art, Mina)	
2 PM	1		(Science, Juan)			
	2	(Math, Carlos)		(Math, Carlos)		
	3	(History, Maria)	(History, Maria)		(History, Maria)	
	4	(Science, Juan)	(History, Wit)			
3 PM	1	(Science, Carlos)	(History, Wit)			
	2					
	3		(History, Maria)			
	4	(Art, Mina)	(Science, Carlos)	(History, Maria)		
4 PM	1					
	2					
	3					
	4					

Table 14: Best Schedule for Scenario 3 with round robin strategy with class indexed.

Hour	Professor ID	Monday	Tuesday	Wednesday	Thursday	Friday
8 AM	Juan	(Science, 2)	(Science, 1)	(Math, 3)		
	Carlos					
	Maria					
	Wit			(History, 1)		
	Sia					
	Mina					
	Lebron	(PE, 3)		(PE, 2)	(PE, 2)	
9 AM	Juan	(Math, 3)	(Science, 2)	(Science, 1)		
	Carlos					
	Maria					
	Wit	(History, 1)				
	Sia	(English, 2)	(English, 1)	(English, 2)	(English, 3)	(English, 4)
	Mina					
	Lebron		(PE, 4)			
10 AM	Juan	(Science, 4)	(Math, 2)	(Math, 1)		
	Carlos					
	Maria					
	Wit		(History, 4)			
	Sia	(English, 3)	(English, 1)			
	Mina					
	Lebron					(PE, 4)
11 AM	Juan					
	Carlos	(Math, 1)	(Science, 2)			
	Maria					
	Wit	(History, 4)				
	Sia		(English, 1)	(English, 3)	(English, 3)	(English, 4)
	Mina	(Art, 2)		(Art, 2)	(Art, 4)	
	Lebron					
12 PM	Juan					
	Carlos	(Science, 2)	(Math, 3)	(Math, 1)		
	Maria					
	Wit					
	Sia			(English, 2)		(English, 4)
	Mina	(Art, 3)	(Art, 2)	(Art, 3)	(Art, 3)	
	Lebron	(PE, 1)	(PE, 1)			
1 PM	Juan	(Science, 4)	(Math, 2)			
	Carlos	(Math, 3)		(Math, 1)		
	Maria		(History, 1)			
	Wit					
	Sia	(English, 1)	(English, 4)	(English, 2)		
	Mina	(Art, 2)	(Art, 3)	(Art, 4)	(Art, 4)	
	Lebron			(PE, 3)		
2 PM	Juan	(Science, 4)	(Science, 1)			
	Carlos	(Math, 2)		(Math, 2)		
	Maria	(History, 3)	(History, 3)		(History, 3)	
	Wit		(History, 4)			
	Sia					
	Mina					
	Lebron					
3 PM	Juan					
	Carlos	(Science, 1)	(Science, 4)			
	Maria		(History, 3)	(History, 4)		
	Wit		(History, 1)			
	Sia					
	Mina	(Art, 4)				
	Lebron					
4 PM	Juan					
	Carlos					