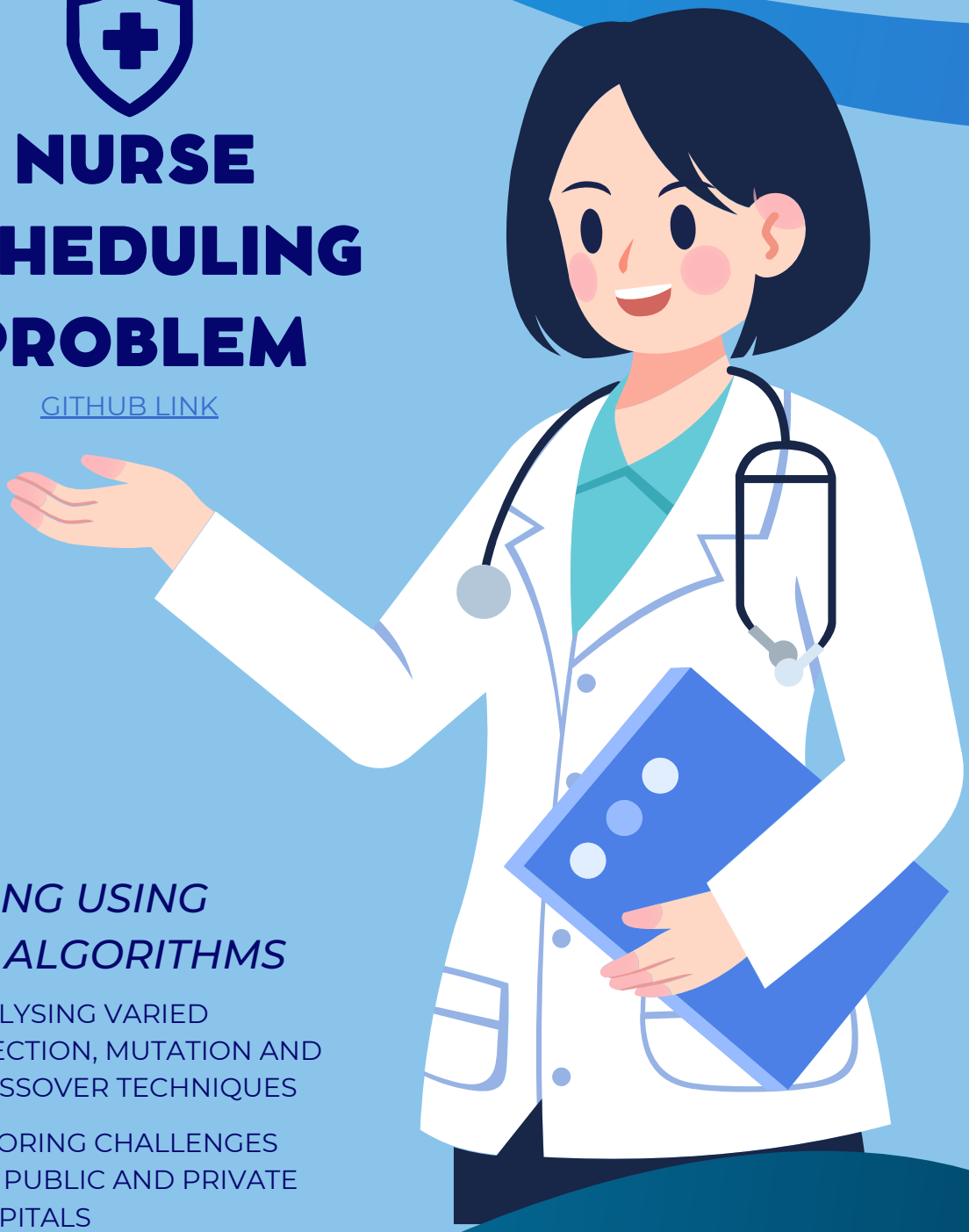
NOVA IMS
Information Management School

NURSE SCHEDULING PROBLEM

GITHUB LINK



OPTIMIZING USING GENETIC ALGORITHMS

- ANALYSING VARIED SELECTION, MUTATION AND CROSSOVER TECHNIQUES
- TAILORING CHALLENGES FOR PUBLIC AND PRIVATE HOSPITALS

Corssino Tchavana Nº 20220597
Eduardo Palma Nº 20221022
Miguel Ramos Nº 20210581
José Matos Nº 20220607

1. Introduction

This report presents a project undertaken as part of the Computational Intelligence for Optimization course within the Data Science and Advanced Analytics Master's program. The objective of this project was to utilize Genetic Algorithms for solving scheduling problems of varying complexities. Our goal was to demonstrate how we applied our understanding of Genetic Algorithms and Optimization Problems to generate the most optimal schedule. The report discusses our implementation approach and provides justifications for our decisions, particularly in relation to the representation of the problem, the fitness function, and the selection, crossover, and mutation methods. Furthermore, we present the results obtained for different problem difficulties and identify the models that exhibited the best results and generalization, thus representing the most effective approaches we discovered.

2. The Problem

To enhance the practicality of the problem, our focus throughout the project was on nurse scheduling within a hospital setting. Specifically, we designated the scheduling problem in a private hospital as representing the easy difficulty level, whereas the scheduling problem in a public hospital represented the medium-high difficulty level. Both problems exhibit common traits, including the division of each day into three eight-hour shifts, a scheduling period lasting 28 days, and a specified number of workers required per shift. However, we introduce variations in other variables such as the number of nurses to schedule, holidays, individual preferences, and skills. In the upcoming sections, we will provide detailed explanations of these variables and explore their impact on the fitness function. Our problem primarily focuses on maximization, but it can also be applied to minimization (although we don't see how it would be useful). It's worth mentioning that our implementation relies on the Charles Library, which was developed as part of our coursework.

2.1. Representation Function

For each difficulty level in the different problems, we will employ distinct representations. The reason for this is that the private hospital has an additional sixth nurse compared to the five in the public hospital. Therefore, the representation needs to account for this extra worker. The representation will function as follows (illustrated schematically below for a simpler case): every 84 indices correspond to a worker (due to the 28 days multiplied by 3 shifts), resulting in 420 indices for the hard difficulty and 504 indices for the easy difficulty. Each of these indices represents a specific shift on a specific day for each nurse. For instance, index 0 represents the first shift of the first day for worker 1, index 2 represents the third shift of the first day for worker 1, index 84 represents the first shift of the first day for worker 2, index 87 represents the first shift of the second day for worker 2, and so on. Our representation is binary, alternating between 1 and 0, indicating whether the nurse works during that shift or not. It is important to note that multiple nurses can work during the same shift.

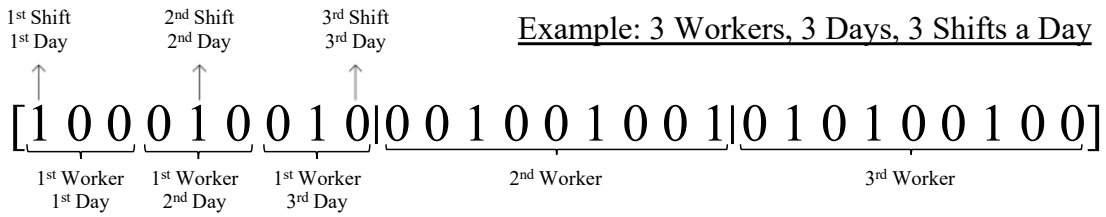


Figure 1. Representation example for a simpler schedule problem

To initiate our representation, we opted to develop a function that effectively distributes the shifts. In other words, this function calculates the total number of nurses required for all shifts based on our data sheet and then evenly assigns the shifts to the nurses without considering any other constraints. This approach avoids a completely random distribution of shifts among the nurses. By implementing this method, we aimed to provide the genetic algorithm with a better starting fitness right from the beginning.

2.2. Fitness Function

As any maximization problem, we decided to give positive fitness to the constraints that we want to happen and negative fitness to constraints that we don't want to happen. In the next table we are going to present an overview of how the different constraints we created impact the fitness of our problem. One of the things we defined is that at least for the schedule to be minimum acceptable the fitness have to reach a positive value.

Constraints	Description	Impact on Fitness
Nurses Distribution	Evaluates whether the allocation of nurses to shifts is evenly distributed, specifically by verifying if each nurse is assigned a number of shifts ranging from 16 to 24 within the 28-day period (pertaining to this particular problem).	If the distribution of nurses' schedules is deemed satisfactory, each nurse who has a well-distributed schedule will impact the fitness by +1000.
2 Shifts in Row	Examines instances where a nurse works consecutive shifts, specifically identifying cases where a nurse is scheduled for two shifts back-to-back.	For every occurrence of consecutive shifts worked by a nurse, the impact on fitness is -100.
3 Shifts in 2 Days	Assesses situations where a nurse is scheduled for more than three shifts within a span of two consecutive days.	Every time a nurse is scheduled for more than three shifts within a span of two consecutive days, the impact on fitness will be -100.
Filled Shifts	Verifies whether the required number of nurses per shift was fulfilled. The relevant data is stored in the sheet named "data.py", where each index corresponds to the number of nurses needed for a specific shift and day. The same data is used in both difficulty levels.	For every occurrence of the shortfall in the number of nurses per shift, the fitness will be impacted by -300.
Skilled Shifts	Examines whether the nurses assigned to shifts that require specific skills possess the necessary qualifications. The relevant data, which varies based on the difficulty level, is stored in the sheet named "data.py."	For every occurrence where a nurse is assigned to a shift without the required skills, the fitness will be impacted by -200.

Constraints	Description	Impact on Fitness
Holidays	Verifies whether any nurse has been assigned to a shift on a day that they have booked as a holiday. This information is stored in "data.py" and varies depending on the difficulty level. The number 1 is used to indicate when a nurse has chosen to take time off during a specific shift and day.	For every occurrence of a nurse being assigned to work on a day they have booked as a holiday, the fitness will be impacted by -100.
Preferences	Verifies whether the preferences of the nurses regarding the shifts they would like to work or not have been accommodated. This information is stored in the sheet named "data.py" and varies depending on the difficulty level. In the sheet, the number 1 indicates that a nurse prefers to work a particular shift, while the number 0 signifies that the nurse has no preference for or against working in that specific shift. On the other hand, the number -1 indicates that the nurse would prefer not to work during that shift.	For every preference that is not fulfilled, the fitness will be impacted by -10. For every preference that is fulfilled, the fitness will be impacted by +10.

Figure 2. Impact of the Different Constraints on the Fitness

The assigned fitness values for each constraint were experimentally tested to determine the most suitable ones for achieving our objectives in relation to each shift. This involved a trial-and-error process to assess and identify the optimal fitness values that aligned with our desired goals for each specific shift. Among the constraints that have a negative impact on fitness, the most critical one is ensuring fully staffed shifts, as it is essential to avoid any shortage of nurses in specific shifts. Thus, this constraint was assigned the highest level of importance. Following the order of importance, the next crucial constraint we considered was the allocation of skilled nurses to specific shifts, as it is vital to have nurses with the appropriate expertise assigned to relevant shifts. Furthermore, we assigned equal and relatively lower importance to the constraints related to consecutive shifts, three shifts in two days, and nurse holiday requests. While these constraints are still relevant, they were deemed less critical compared to fully staffed shifts and skilled assignments. Regarding preferences, we utilized them as tiebreakers when two solutions were otherwise equal in meeting the constraints. Their role was to prioritize individual nurse preferences and determine the preferred solution.

It is important to consider that the data used varies depending on the difficulty level. The primary differences lie in the number of holidays granted and the presence of preferences and skilled shifts. In the medium-hard difficulty level, nurses are entitled to two holiday days per week, whereas in the easy difficulty level, they have one holiday day per week. Additionally, the easy difficulty level has fewer preferences and skilled shifts compared to the medium-hard level. Furthermore, it is worth noting that holidays are granted for full days only. In other words, nurses cannot take holidays for specific shifts; it must be for a consecutive set of three shifts. Additionally, on weekends, both Saturdays and Sundays, there is only the second shift available.

2.3. Selection Methods

Regarding the selection methods employed, we attempted to address our problem by utilizing three different approaches: Fitness Proportionate Selection, Tournament Selection, and Ranking. In this section, we will focus on explaining the specific modifications made in

comparison to the practical classes, aiming to keep this report more practical than theoretical. The Tournament Selection implementation remained identical to the one conducted during the practical class. However, some modifications were made to the Fitness Proportionate Selection approach. In the case of maximization, where individuals can have both positive and negative fitness values, we encountered an issue when the total fitness of the population reached zero. This posed a challenge when generating a random number between 0 and the total fitness. To tackle this problem, we devised a solution: we augmented each fitness value by the absolute value of the lowest fitness value in the entire population. By implementing this adjustment, the lowest fitness value within the population became 0, while maintaining positive values for all other fitness values. Subsequently, the implementation proceeded as usual. Next, we needed to adapt the implementation for a minimization problem, as we wanted our model to be capable of handling both maximization and minimization scenarios. The approach for minimization is the reverse, as we aim to assign higher probabilities to individuals with lower fitness values. To achieve this, we altered the fitness calculation. Instead of using the actual fitness value, we computed it as the subtraction between the individual with the highest fitness in the population and the individual's fitness that we aimed to evaluate. This value was then divided by the sum of the individual with the highest fitness in the population and a very small number (0.0001). The subsequent steps of the implementation were similar to those of the maximization problem. The code contains the detailed implementation.

Finally, the Ranking Selection Method was implemented from scratch and it involves assigning ranks to individuals in the population based on their fitness values and then selecting individuals for reproduction based on their ranks. It provides a way to balance the selection pressure across the population, giving a chance for less fit individuals to be selected while still favouring individuals with better ranks. Very good at maintaining the diversity in the population.

2.4. Crossover Methods

In relation to the crossover methods employed, we decided to address our problem by utilizing four distinct approaches: Single Point Crossover, Cycle Crossover, Uniform Crossover, and Partially Mapped Crossover. While the implementations for Single Point Crossover closely resembled the one conducted in the practical class, there were some variations in the steps taken for Cycle Crossover and Partially Mapped Crossover. Due to our binary representation, implementing Cycle Crossover and Partially Mapped Crossover as in the class proved challenging, as we had repeated occurrences of 0s and 1s in the representation. To overcome this limitation, we adopted an alternative approach. We utilized the indexes of the parents, which were always associated with the values they represented. These indexes were shuffled, and the regular Cycle Crossover and Partially Mapped Crossover operations were performed. Subsequently, the indexes of the offspring were reordered, and they were converted back into the original binary representation, with the associated 0s and 1s.

Lastly, the Uniform Crossover Method was developed from the ground up. This method operates by randomly selecting genes from the parents to exchange between them. Each offspring has an equal chance of inheriting a specific gene from either parent.

2.5. Mutation Methods

In relation to the mutation methods employed, we sought to tackle our problem by utilizing three distinct approaches: Binary Mutation, Swap Mutation, and Inversion Mutation. The implementations for all these methods closely resembled those presented in the practical classes.

3. Statistical Analysis

3.1. Best Combination of Methods

To determine the optimal combination of methods, we conducted a grid search for both difficulty levels. This involved trying out every possible combination of our selection, crossover, and mutation methods while keeping the crossover and mutation probabilities fixed at 85% and 15%, respectively. These probability values were chosen based on existing literature in the field. It is worth noting that we excluded the Cycle Crossover method from our evaluation as it consistently yielded inferior results compared to other crossover methods during our testing phase. For this specific problem, the outcomes indicate that the Uniform Crossover is the most effective crossover method, while the Ranking Selection is the optimal selection method. Moreover, both Binary Mutation and Swap Mutation result in favourable fitness values. Specifically, the easy level achieved fitness values of 1571 and 1261, while the medium-hard level obtained values of 598 and 146. Elitism was used in both cases (better results), 35 repetitions were performed, and the initial population size was 100. All the possible combinations are presented in the "Statistical Analysis" Jupyter Notebook.

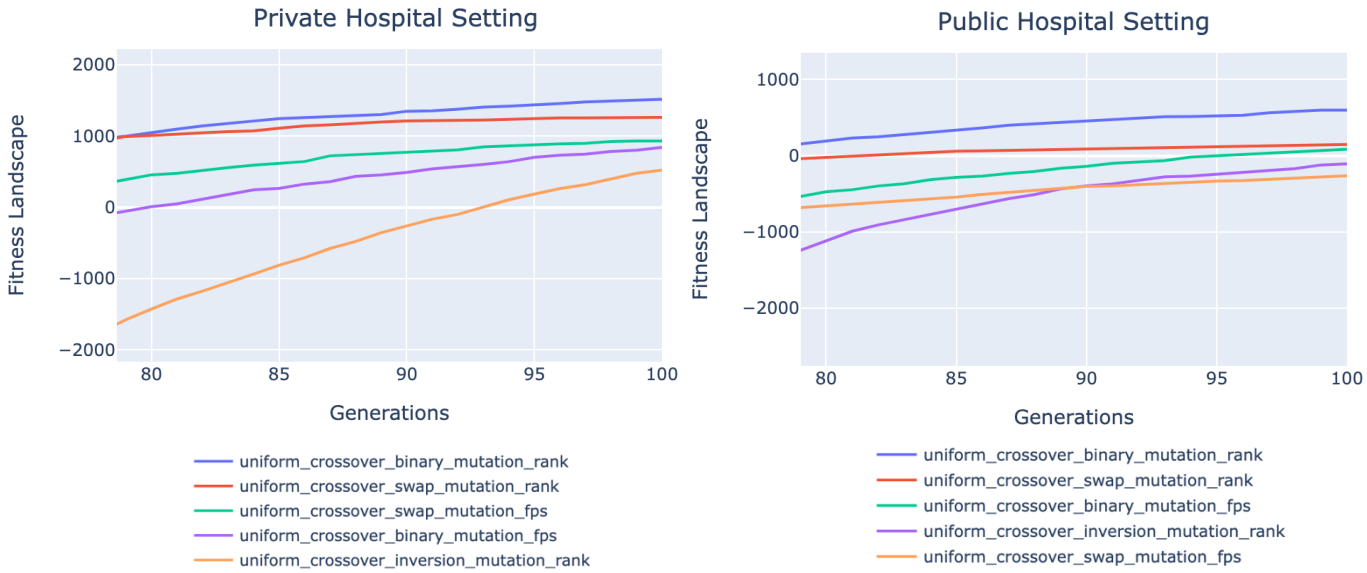


Figure 3. and 4. Line Chart for Best Combination of Methods for the Private and Public Hospital Setting

3.2. Crossover and Mutation Probabilities Fine-Tuning

Subsequently, we conducted an additional grid search to determine the optimal crossover and mutation probabilities for the two most effective methods identified earlier. During the grid search, we varied the mutation method between Binary Mutation and Swap Mutation, while adjusting the crossover probability and mutation probability in increments of 20 percentage points. In the case of the easy difficulty problem, the optimal combination was found to be Swap Mutation with a 100% crossover probability and an 80% mutation probability, resulting in a fitness score of 2304. As for the medium-hard difficulty problem, the best combination involved Binary Mutation, a 100% crossover probability, and a 60% mutation probability, resulting in a fitness score of 1252. Elitism was used in both cases (better results), 35 repetitions were performed, and the initial population size was 100. All the possible combinations are presented in the "Statistical Analysis" Jupyter Notebook.

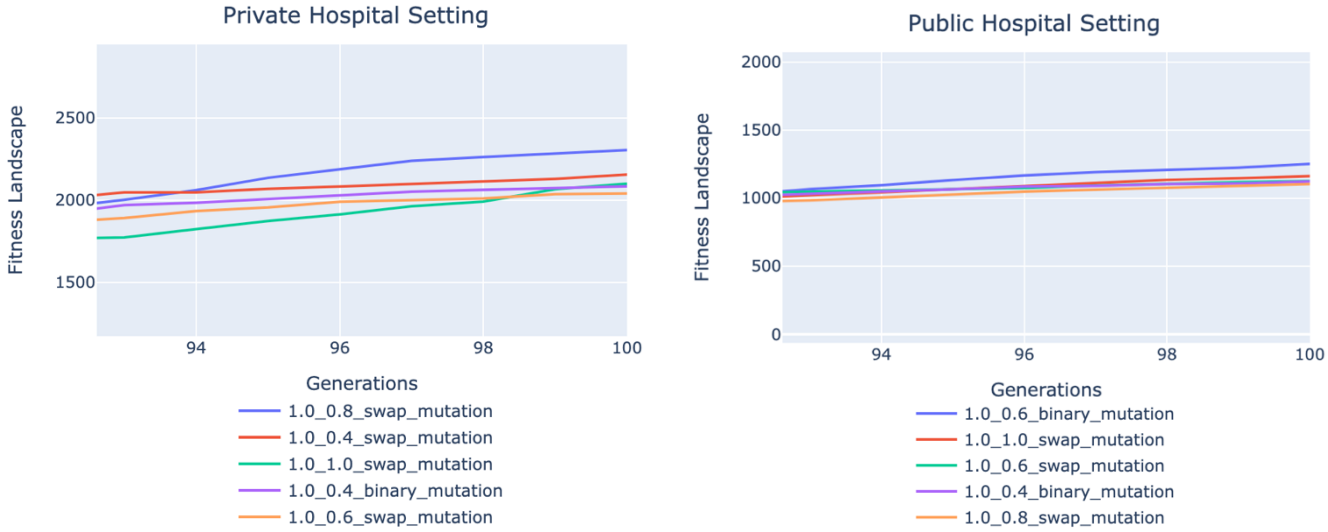


Figure 5. and 6. Line Chart for Best Combination of Crossover and Mutation Probabilities

4. Conclusion

As the best methods for both scenarios are remarkably similar we chose to adopt a definitive combination of methods that we believe exhibits superior generalization across all difficulty levels. This preferred approach entails employing Ranking Selection, Uniform Crossover with a probability of 100%, and Swap Mutation with a probability of 100%. By implementing this combination over the course of 200 generations we undeniably attain remarkable outcomes for both difficulty levels, as demonstrated below. As a result, we are confident that the project's objectives have been successfully achieved. However, one area that could be enhanced is the consideration of fitness value constraints. Currently, the constraints were solely designed with the Public Hospital Setting in mind, overlooking the importance of incorporating the Private Hospital Setting as well.

	Private Hospital	Public Hospital
Fitness	3820	2420
Distributed Workers	6	5
Consecutive Shifts	4	5
Shifts in 2 Days	0	0
Shifts Not Completely Filled	4	1

	Private Hospital	Public Hospital
Holidays Not Respected	3	9
Number of Failed Skilled Shifts	1	3
Number of Preferences Respected	41	74
Number of Preferences Not Respected	49	102

Figure 7. Possible Final Results for Each one of the Problems