COMPUTATIONAL INTELLIGENCE FOR OPTIMIZATION

FROZEN LAKE PROBLEM

USING GENETIC ALGORITHMS

MAY 2023

ALEX SANTANDER	20220658
CARLOTA CARNEIRO	20210684
INÊS ROCHA	20220052
SUSANA DIAS	20220198

NOVA IMS Information Management School

INTRODUCTION

This report focuses on solving the Frozen Lake Problem (FLP) using Genetic Algorithm (GA) techniques. For the development of the project, we implemented and evaluated various selection, crossover, and mutation methods within the framework of GA for problem-solving. The main objective of this was to assess their performance, convergence, and effectiveness and compare their efficacy in achieving optimal solutions.

As part of this project, we have developed six files to address various aspects of the problem. The first file, *charles.py*, serves as a foundation for implementing individuals and populations, and facilitates the evolutionary process through selection, crossover, and mutation. The *FL_fitness.py* file is focuses on defining the fitness function for the FLP using the Gym library. It encompasses the setup of the environment, initialization of the state, and simulation of an individual's actions within the environment based on their representation. The fitness function will be explained further in this report. In the *selection.py* file, we have implemented three distinct selection algorithms, namely Tournament, Roulette and Rank. These algorithms play a crucial role in determining the individuals that proceed to the next generation based on their fitness. The *crossover.py* is the file where we implemented six crossover algorithms – Arithmetic, Single Point, Multi-Point, Uniform, Uniform Mask and Heuristic. These algorithms enable exchange of genetic information between individuals during the reproductive process, contributing to the exploration of different genetic combinations. Similarly, in the *mutation.py* file, we have incorporated three mutation algorithms, namely Scramble, Swap and Inversion. These algorithms introduce diversity into the population by altering the genetic material of selected individuals. Lastly, the *main.py* file conducts experiments by applying various configurations of genetic algorithms to solve the FLP. It initializes a population of individuals, guides their evolution over a specified number of generations using selection, crossover, and mutation operations, and tracks the fitness values of the evolving population. These experiments provide insights into the performance of different algorithmic configurations in tackling the FLP. For further details and access to our project, visit the following GitHub repository: https://github.com/inesleonidasrocha/CIFO2023-Frozen-Lake.

The division of labour within our group was equitable, with each member contributing equally to research, code development, testing, and report creation.

METHODOLOGY

In the process of designing our fitness function, we initially experimented with a simpler approach that involved penalizing the agent with a fixed penalty of -100 if he fell into a hole. However, this approach did not consider the actual path discovery aspect of the problem. We also explored an alternative approach where the penalty decreased progressively as the agent got closer to the goal, coupled with rewarding the agent upon reaching the goal. However, this method did not fully meet our requirements and did not provide the desired outcomes. As a result, we decided not to utilize either of these approaches in our fitness function. Instead, we designed a fitness function for the Frozen Lake problem to evaluate the performance of a solution based on the state of each step taken, considering both valid movements (reaching the goal or not reaching the goal) and invalid movements (falling into a hole).

Valid Movements: When the agent makes a valid movement, there are two possible outcomes: reaching the goal or not reaching the goal. Each valid movement is rewarded with a fitness score of -1. However, there is a special case when the agent is at the edges of the grid, and the movement given would lead the agent outside the grid. In this case, the agent remains in the same spot, and no movement is performed, resulting in a fitness score of -2. The punishment for this action is higher because the goal is to find solutions that reach the goal in the fewest number of steps. In the case where the agent runs out of movements without reaching the goal or falling into a hole, it is assigned a fitness score of -16. Another possibility is the agent running out of movements while staying in the same position for the entire scenario, which results in a fitness score of -32.

Invalid Movements: The upper limit for this section is set based on the lowest fitness value for valid movements, which is -32. It is logical to assume that the best fitness among the worst solutions (falling into a hole in the last movement) is slightly lower than -32. Therefore, the fitness value for the best of the worst solutions is assigned as -33 and referred to as the "Baseline Punishment".

Repetition of Positions: It is important to note that the best of the worst solutions is the one that does not repeat the same position in any movement. If any repetition occurs, a penalty of -1 is added to the baseline punishment for each repeated position. This ensures that a failed solution with 14 valid movements is considered better than a solution with 13 valid movements but with repetitions.

Baseline Values: To distinguish between solutions with different numbers of valid movements and repetitions, a set of 16 values is used between the baseline punishments – number of spaces on a 4x4 grid. For instance, if the baseline punishment for a solution with 14 valid movements is -40, the next baseline punishment for a solution with 13 valid movements (without repetition) is -56. This ensures that a solution with 14 valid movements and repetitions in the same position is always considered better than a solution with 13 valid movements without repetition.

Best Fitness: The optimal fitness in our problem is attained by adhering to the methods, punishments, and rules outlined previously. The global optimum is reached when the agent successfully reaches the goal with the fewest possible movements, without falling into a hole or repeating any movements. This achievement corresponds to a fitness value of -5, as we penalize the agent with -1 for each step taken and do not provide any additional reward for reaching the goal. Hence, we can conclude that our defined fitness function treats the FLP as a maximization problem.

By designing the fitness function in this manner, we encourage the GA to discover solutions that efficiently reach the goal, avoid falling into holes, and minimize position repetitions. Our problem is represented by a vector of size 16, chosen to match the dimensions of the 4x4 grid in the FLP. Each value in the vector corresponds to an action taken by the agent in the FLP. We opted for a vector of size 16 because it provides ample capacity to navigate the grid without getting stuck in an infinite loop or failing to reach a solution.

CONFIGURATION TESTS & DISCUSSION

CHOSING THE CROSSOVER

Our approach to solving the FLP involved experimenting with various combinations of selection, crossover, and mutation operators. Specifically, we conducted these experiments using a population size of 50 individuals and ran the evolutionary process for 100 generations, we set the replacement parameter to True, and in this first stage of the analysis, we assumed a non-slippery ice condition, meaning that the probability of sliding on the ice was not a factor in our simulations.

During our initial analysis and testing, we excluded the cycle and the partially mapped crossover operator – pmx - methods that were taught in class. This decision was based on the nature of our problem, where our individuals consist of 16 numbers within a valid range of 0 to 3. Since there are repeated numbers within our individuals, the cycle and pmx methods, which rely on the positions of each number for exchanging information between parents, are not suitable for our problem. One possible approach to address this issue would be to consider the individuals on a per-index basis rather than focusing on their values. However, we decided to explore alternative crossover methods that would be better suited to our specific problem.

We conducted experiments on six different crossover methods in our baseline research since we only wanted to focus on three specific methods for our further experiments. To compare the crossover methods and determine which ones achieved the best fitness in the fewest generations, we plotted a fitness landscape of all crossover methods using the tournament selection method and the inversion mutation. By analysing Annex 1, which represents the fitness function without elitism enabled, the arithmetic

method was the only one that reached the global optimum. The single point, the heuristic and the multi point (number of points set to 5) crossover methods came closest to the global optimum. On the other hand, the uniform mask and uniform crossover methods did not yield good results. This could be attributed to the 0.5 probability used for exchanging information between parents. When we enabled elitism and examined Annex 2, we noticed a significant change in the behaviour of the crossover methods. It became evident that the arithmetic and heuristic methods achieved the global optimum in less than 21 and 60 generations on average, respectively. It is important to notice that enabling elitism leads to the heuristic method also reaching the global optimum, whereas this was not the case when elitism was disabled. This observation suggests that with elitism enabled, only the top-performing individuals are carried forward to the next generation, potentially enhancing the chances of reaching the global optimum. Among the remaining crossover methods, three of them, namely the uniform mask, uniform, and single point methods, achieved the second-best fitness level of -8. We observed a notable difference in the performance of the uniform and uniform mask methods when elitism was enabled or disabled. Taking this into consideration, we concluded that the single point, arithmetic, and heuristic crossover methods were the better choices for our research.

METHODS COMPARISON

In order to evaluate and compare the performance of different selection, crossover, and mutation methods, we conducted an analysis of their average and standard deviation while keeping all other methods constant. We kept the parameters the same as before, that is, using a population size of 50 individuals, 100 generations, the replacement parameter as True, and in this first stage of the analysis, we assumed a non-slippery ice condition. For the selection method (Figure 1), we applied all three selection methods with the arithmetic crossover and inversion mutation. We observed that all three methods converged to the global optimum, but the rank selection method demonstrated the least deviation and reached the global optimum more frequently, on average, and in fewer generations. On the other hand, the tournament selection method exhibited a relatively higher deviation compared to the other methods. Moving on to the crossover methods (Figure 2), we examined the three previously chosen methods with rank selection and inversion mutation. The arithmetic crossover method continued to outperform the others, reaching the global optimum in fewer generations. The single-point crossover method had the poorest fitness performance and also exhibited a higher deviation. Regarding the mutation methods (Figure 3), when using rank selection and arithmetic crossover, we found that all tested methods performed relatively similarly in the initial generations. The scramble mutation method was the first to reach the global optimum, followed by the inversion method, and finally the swap mutation method.

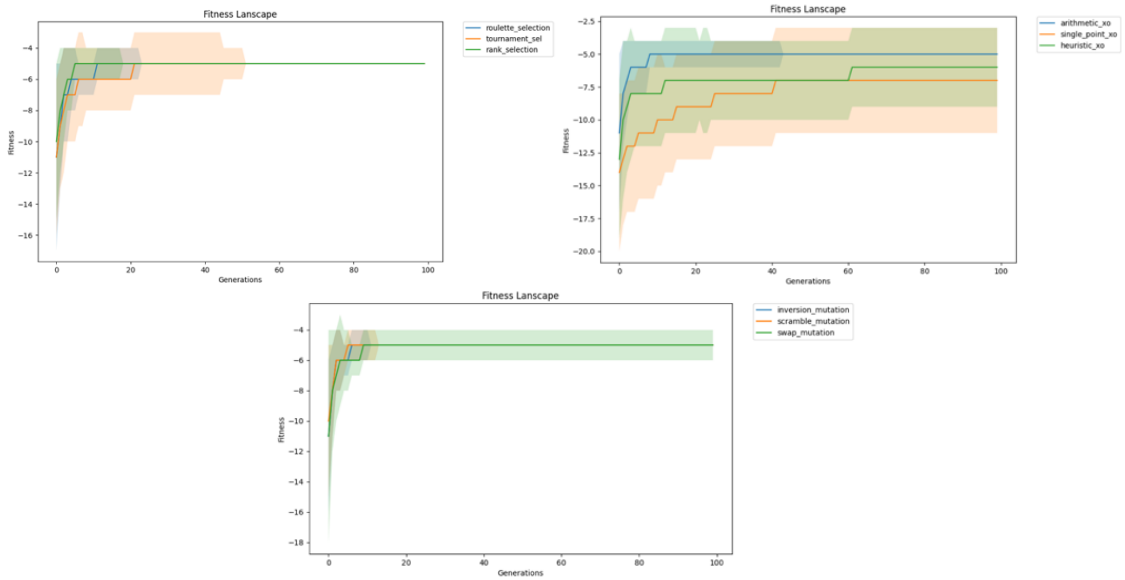


Figure 1 - Selection Methods Comparison without the Slippery Ice Condition enabled (left hand-side)

Figure 2 - Crossover Methods Comparison without the Slippery Ice Condition enabled (right hand-side)

Figure 3 - Mutation Methods Comparison without the Slippery Ice Condition enabled (below)

To assess the impact of the slippery ice condition on our selection of methods, we conducted further analysis by plotting the same graphs as before, this time with the slippery ice condition enabled. In Figure 4, we observe that the rank selection method continues to reach the global optimum in fewer generations, just like in the absence of the slippery ice condition. It is followed by the tournament selection and the roulette selection methods. Similarly, in Figure 5, the arithmetic crossover outperforms the single point and heuristic crossovers in reaching the global optimum in fewer generations, mirroring the behaviour observed without the slippery ice condition (Figure 2). The mutation methods, as shown in Figure 6, also perform well, reaching the global optimum in less than 20 generations on average. Although the three methods exhibit similar behaviour, the swap mutation method converges to the global optimum at a later stage.

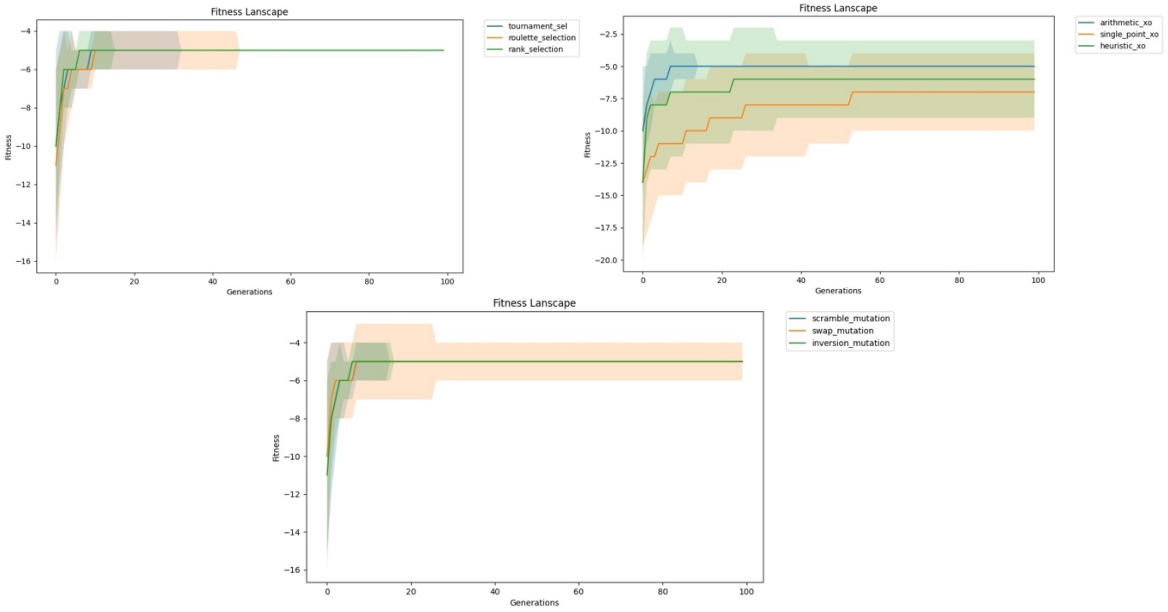


Figure 4 - Selection Methods Comparison with the Slippery Ice Condition enabled (left hand-side)
Figure 5 - Crossover Methods Comparison with the Slippery Ice Condition enabled (right hand-side)
Figure 6 - Mutation Methods Comparison with the Slippery Ice Condition enabled (below)

EXPERIMENTS

With this in mind, we tried to solve the FLP with different combinations of selection, crossover and mutation techniques, always with a population of 50, 100 generations, a crossover probability of 0.9, a mutation probability of 0.2 and always with replacement. We also tried with the elitism enabled and disables, and with the slippery ice condition enabled and disabled. It is important to refer that we attempted to solve the problem with different probabilities for the crossover and the mutation, however, the results were very similar, hence we retained the same probabilities throughout all 108 experiments.

When examining all 108 experiments presented in Table 1, in the Annexes, it becomes evident that the global optimum of -5 was exclusively achieved when utilizing the arithmetic crossover method. Among the 36 experiments conducted using tournament selection, the global optimum was reached solely when employing the scramble and inversion mutation methods. Specifically, all four combinations of elitism and slippery ice conditions, in conjunction with the inversion mutation, resulted in the attainment of the global optimum. However, when utilizing the scramble mutation, the global optimum was only achieved when the slippery ice condition was enabled. Concerning the 36 experiments involving roulette selection, the global optimum was exclusively reached when the elitism condition was enabled. In terms of the scramble and inversion mutation methods, the global optimum was attained under all conditions with elitism enabled, regardless of the presence of the slippery ice condition. Conversely, when employing the swap mutation method, the global optimum was only achieved when the slippery ice condition was disabled. Lastly, in the rank selection experiments, only the experiment utilizing the inversion mutation method without both the elitism and slippery ice conditions enabled failed to reach the global optimum.

After considering all the experiments and the comparisons made between selection, crossover, and mutation methods, it can be concluded that our optimal model consists of rank selection, arithmetic crossover, and scramble mutation. These methods demonstrate the ability to converge towards the global optimum at an early stage. Furthermore, since the inversion mutation yields similar results to the scramble mutation, as observed in figure 3, and also leads to the attainment of the global optimum, it is considered a highly promising model as well. Therefore, we aim to further analyse the performance of these models by exploring variations in the elitism and slippery ice conditions.

In figure 7, we can see the fitness landscape of one of our best models, the rank selection with the arithmetic crossover and scramble mutation, with all the combinations of the elitism and the slippery ice conditions. We can see that in this experiment, when the elitism was disabled and the slippery ice enabled, the global optimum wasn't reached. Additionally, we can understand that when the elitism was enabled, it reached the global optimum in the same generation, independently of the slippery ice condition.

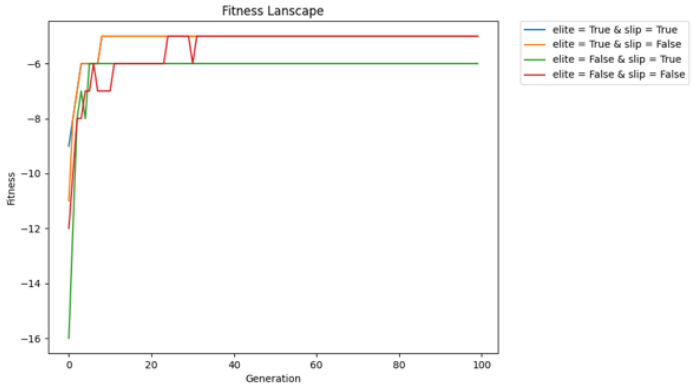


Figure 7 - Rank Selection, Arithmetic Crossover and Scramble Mutation

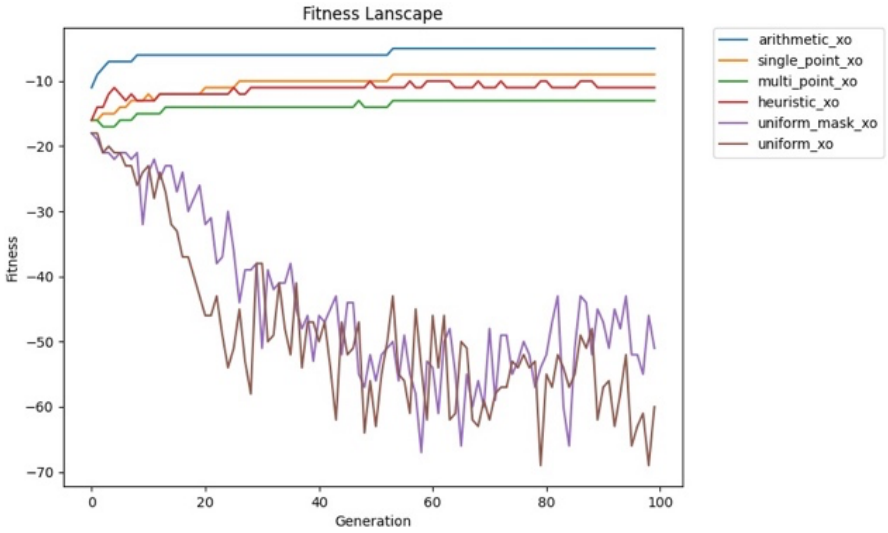
CONCLUSION

This project has provided us with valuable insights into the usage of GAs for optimization problems, specifically the FLP. Through our self-study, we have gained a deeper understanding of GAs and discovered new selection, crossover and mutation methods not covered in our class. By implementing various combinations of methods to solve the FLP, we believe that we have obtained promising results, with the slippery condition both enabled and disabled in the environment. In our experimentation with selection, crossover, and mutation methods (as depicted in Figures 1 to 6), most of the tested methods achieved the global optimum in just a few generations. However, it is worth noting that only the arithmetic crossover method consistently reached the global optimum. This suggests that further investigation into alternative crossover methods that perform better for this type of problem would be beneficial. Nonetheless, it is important to highlight the significant improvement in performance achieved with the arithmetic crossover method, which presented a positive impact in our models – only the models with this type of crossover reached the global optimum.

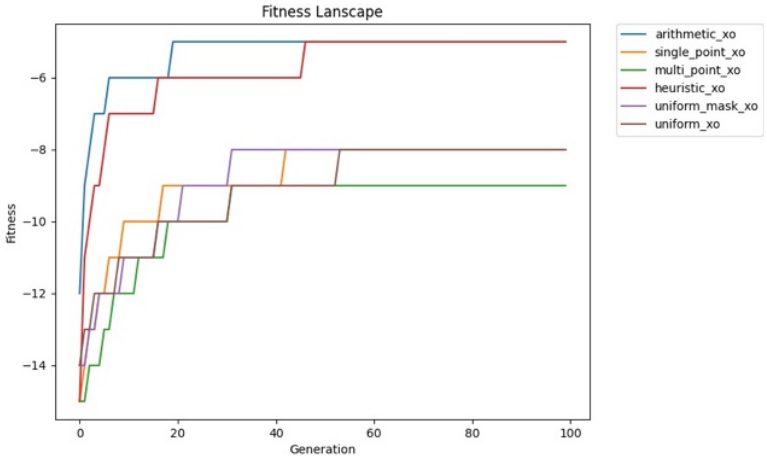
Furthermore, we recognize the potential for testing different combinations, such as varying types of elitism, to further explore and refine the performance of our models. Based on our analysis and conclusions thus far, we cans say that different operators significantly influence the convergence of the GA, as well as elitism.

ANNEXES

ANNEX 1 - Fitness Landscape of the Crossover Methods (Elitism False)



ANNEX 2 - Fitness Landscape of the Crossover Methods (Elitism True)



ANNEX 3 – Combination of the Experiments and Results for the Frozen Lake Problem with Genetic Algorithms

Combination of the Experiments and Results for the Frozen Lake Problem with Genetic Algorithms

Selection	Crossover	Mutation	Elitism	Slippery	Average Fitness
Tournament	Arithmetic	Swap	True	True	-6
				False	-6
			False	True	-6
				False	-6
		Scramble	True	True	-5
				False	-6
			False	True	-5
				False	-6
		Inversion	True	True	-5
				False	-5
			False	True	-5
				False	-5
	Heuristic	Swap	True	True	-7
				False	-7
			False	True	-14
				False	-14
		Scramble	True	True	-6
				False	-6
			False	True	-12
				False	-12
		Inversion	True	True	-6
				False	-6
			False	True	-13
				False	-12
	Single Point	Swap	True	True	-12
				False	-11
			False	True	-12
				False	-11
		Scramble	True	True	-10
				False	-9
			False	True	-11
				False	-10
		Inversion	True	True	-10
				False	-9
			False	True	-10
				False	-9
Roulette	Arithmetic	Swap	True	True	-6
				False	-5
			False	True	-7
				False	-8
		Scramble	True	True	-5
				False	-5
			False	True	-7
				False	-7
		Inversion	True	True	-5
				False	-5
			False	True	-7
				False	-6
	Heuristic	Swap	True	True	-7
				False	-8
			False	True	-23
				False	-22
		Scramble	True	True	-7
				False	-6
			False	True	-22
				False	-21
		Inversion	True	True	-6
				False	-6
			False	True	-23
				False	-21
	Single Point	Swap	True	True	-11
				False	-11
			False	True	-19
				False	-19
		Scramble	True	True	-11
				False	-10
			False	True	-19
				False	-19
		Inversion	True	True	-10
				False	-12
			False	True	-19
				False	-19
Rank	Arithmetic	Swap	True	True	-5
				False	-5
			False	True	-6
				False	-6
		Scramble	True	True	-5
				False	-5
			False	True	-5
				False	-5
		Inversion	True	True	-5
				False	-5
			False	True	-5
				False	-5
	Heuristic	Swap	True	True	-7
				False	-7
			False	True	-25
				False	-25
		Scramble	True	True	-6
				False	-6
			False	True	-24
				False	-25
		Inversion	True	True	-6
				False	-7
			False	True	-24
				False	-25
	Single Point	Swap	True	True	-10
				False	-11
			False	True	-15
				False	-13
		Scramble	True	True	-8
				False	-9
			False	True	-13
				False	-13
		Inversion	True	True	-10
				False	-9
			False	True	-13
				False	-13