NOVA IMS Information Management School

COMPUTATIONAL INTELLIGENCE FOR OPTIMIZATION

MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS

Genetic Algorithm for Frozen Lake Problem

`github clone: https://github.com/rafdinis/FrozenLake.git`

Group 18

Rafael Dinis, number: 20221643

Ricardo Arraiano, number: 20210995

Sabeen Mubashar, number: 20220726

Stanislav Slesarev, number: 20221396

28 May, 2023

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

github link: **https://github.com/rafdinis/FrozenLake**

Abstract

Our objective is to apply genetic algorithms to solve an optimization problem, specifically the Frozen Lake problem. We utilized a 4x4 matrix and experimented with various selection algorithms (Tournament, Ranking, and Roulette), different crossover methods (Single Point, Double Point, and Uniform), and Mutation Operators (Random Resetting, Swap, and Scramble) with different rates and parameters. To evaluate the performance, we conducted a grid search using three different fitness functions based on Euclidean distance, Manhattan distance, and Radial distance. Among the tested configurations, we found that Roulette wheel selection algorithm, Single Point crossover, and scramble mutation operator using Euclidean distance as the fitness function gave the best results.

1. Introduction

The Frozen lake problem consists in the interaction between an agent and the environment where the goal is to arrive at the final destination alive. The environment is a grid where we have safe positions and holes, if the agent finds himself in a hole he loses the game. The agent can choose between four different actions , moving up, down, right or left and the point is to maximize the chance of arriving at the goal.

We are trying to solve this problem with the help of genetic algorithms, where we initialize the population and by choosing the best combination of selection, crossover and mutation algorithms over the different generations we try to arrive to the global optimum, of the maximization problem, which corresponds arriving to the goal, fitness equals one, as fast as possible.

2. Methodology

2.1 Environment

The environment is implemented using the OpenAI Gym library to create and interact with the environment class for the frozen lake problem. It has three parameters: *env_name* which specifies the environment we opted for the 4x4 matrix, and *is_slippery* indicates whether the ice is slippery or not, as well as *render_mode*, that allows us to see the agent in the environment when we set it to True. We tested the environment for both slippery and not slippery .

2.2 Agents

The class called 'Agent' represents an agent with a policy_matrix of size 4x4 is generated with values ranging from 0 to 3 and each value represents an action 0 for left, 1 for down,2 for right and 3 for up and determined what action to take.The default distance matrix is Manhattan distance while other distance metrics are radial and Euclidean.

2.3 Genetic Algorithm

2.3.1 Selection

We implemented three selection algorithms Tournament Algorithm, Ranking Selection Algorithm, and Roulette Wheel Selection Algorithm. Initially, all selection algorithms were implemented without elitism, but it was later decided to incorporate elitism to preserve the best individuals in each generation. For the *Tournament Algorithm*, the tournament size was initially set to 2, but it was increased to 4 when elitism was introduced for more comparisons and increasing the probability of selecting fitter individuals and striking a balance between exploration and exploitation while maintaining computational efficiency. .After testing different elitism rates and different selection algorithms, it was found that the roulette algorithm outperform all other algorithms with an elite rate of 0.2.

2.3.2 Crossover

In the context of our problem, we decided to implement the following crossovers - Single point, Double Point and Uniform. Considering the probability of a crossover and having in mind this probability should be high and close to 1, we experimented with different values and picked rate=0.8 for our final solution.
Figure 3.1 shows the practical application of crossovers to the policy matrix of our agents.

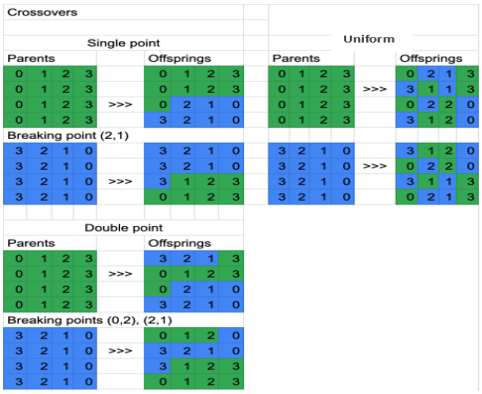

Figure 3.1 - Crossovers visualization

2.3.3 Mutation

In the context of our problem, we decided to implement the following mutations - Random Resetting, Swap and Scramble. Considering the mutation rate and having in mind this rate should be low and closer to 0, we experimented with different values and picked rate=0.2 for our final solution. The values lower than 0.2 seemed to significantly slow down the overall process of convergence. Figure 3.2 shows how in practice the way every mutation changes the policy matrix of our agent.

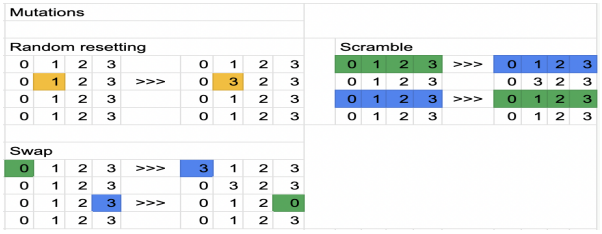

Figure 3.2 - Mutations visualization

2.4 Grid Search

Grid search is a systematic way to explore all the possible combinations of different processes and operators. The algorithm should be executed for every unique combination, which allows one to find the optimal or near-optimal combination for a given task.

Figure 3.3 shows the actual grid search visualization of all combinations.

Grid search combinations								
000	001	002	010	011	012	020	021	022
100	101	102	110	111	112	120	121	122
200	201	202	210	211	212	220	221	222

Figure 3.3 - Grid search Visualization

3. Results

Fitness functions are used to evaluate the fitness of different individuals in our population. We used three different fitness functions, one based on Euclidean distance, other on Manhattan distance and the last one on Radial distance. We wanted to test and evaluate the differences when using Slippery = True and when Slippery = False with runs=100 and gens=500. We proceed to a grid search in order to find the best combinations of selection crossover and mutation rates.

Top 5 results for each metric			
Non-slippery	manhattan	euclidian	Radial
202	0.889	0.919	0.85
211	0.886	0.925	-
212	0.879	0.889	-
201	0.873	-	0.86
221	0.87	0.896	0.888
O12	-	0.872	-
222	-	-	0.869
OO1	-	-	0.845

Top 5 results for each metric			
Slippery	manhattan	euclidian	radial
210	0.676	0.734	0.675
200	0.615	0.652	0.643
220	0.601	0.749	0.683
212	0.547	0.531	0.49
211	0.51	0.521	0.535

Colors in table	
	Best out of top 5
	Second Best
	Worst out of top 5

The best Results are obtained with the Euclidean Distance both for Non-slippery and Slippery. As expected the results are worst when Slippery is True. The performance of combination 202(Roulette wheel, single point crossover, scramble mutation operator) and 220(Roulette wheel, uniform crossover and scramble) seems to outperform the others respectively. The main difference between the three metrics in the performance of our GA is that the fitnesses values of the algorithms decreases from euclidean to manhattan and manhattan to radial in Non-slippery and from euclidean to radial and from radial to manhattan in slippery.

Algorithm Indexes			
Index	**Selection**	**Crossover**	**Mutation**
0	Tournament	Single Point	Random Resetting
1	Ranking	Double Point	Swap
2	Roulette Wheel	Uniform	Scramble

The most optimal outcome is achieved using the Slippery is false, which shows a median of 100% fitness, a maximum fitness of 91.9%, and an 88% success rate. This result is obtained

through Roulette wheel selection algorithm with an elite rate of 0.2, a single point crossover rate of 0.8, and the scramble mutation operator. We subsequently implemented this combination into the main1.py file to optimize frozen lake problem.

4. Discussion

The top 5 algorithms for the more robust manhattan and euclidean fitnesses all performed between 84.0% and 92.5%, whereas the median fitness was 100.0% and the success rate was between 77.0% and 88.0%.

There is opportunity to improve these algorithms, including by implementing a NEAT approach to allow the agent to have a direct input from the environment (whether its immediate surrounding or the full map). The code could also be refined to become size-independent and to be able to perform in other map sizes. Another point of improvement is applying multi-objective optimization techniques on the fitness function, for instance by adding a secondary fitness that is proportional to the overall distance to holes (similar to our experience of vertigo) or a secondary fitness that promotes the shortest path to the goal.

5. Conclusion

With this project the main goal was to develop a genetic algorithm in order to find a optimal solution for our problem, get an agent to walk in a lake without falling in the wholes and arriving to the final destination.

We believe that the capacity of combining different metrics and selection, crossover and mutation rates makes the genetic algorithms a very strong solving problem for maximization or minimization solutions, being our problem a minimization one. Without the need of analyzing the entire population of possible candidates the GA and its growing with time make it possible to converge faster, be more precise and be less computationally expensive than other algorithms where analyzing the entire population is mandatory.

To conclude we think this opportunity to work and learn about GAs was very rich and useful for future tasks.

References:

1. Vanneschi, L., & Silva, S. (2023). *Lectures on Intelligent Systems*. Springer Nature. https://doi.org/10.1007/978-3-031-17922-8
2. De Waele, M., & Nowe, A. (2008). Uniform Crossover in Genetic Algorithms. In Proceedings of the 4th International Conference on Natural Computation, August 18-20, 2008, Jinan, China (pp. 769-773). Retrieved from https://www.researchgate.net/publication/201976488_Uniform_Crossover_in_Genetic_Algorithms
3. Gym Library. (n.d.). FrozenLake-v0. Retrieved May 28, 2023, from https://www.gymlibrary.dev/environments/toy_text/frozen_lake/
4. Tutorialspoint. (n.d.). Genetic Algorithm - Quick Guide. Retrieved May 28, 2023, from https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_quick_guide.htm

Overall best result.