Evolutionary Computation

Ricardo Guegan[2020211358] and Guilherme Almeida[2019224555]

Universidade de Coimbra

Abstract. Evolutionary Computation in Path Finding: This paper presents an in-depth analysis of evolutionary algorithms applied to the Frozen Lake Problem, aiming to discover the most efficient path to a goal. The authors explore two distinct representations string of actions and map of actions.

Keywords: Evolutionary Computation · Path finding · Frozen Lake Problem

1 Introduction

In this report we will explain how we used evolutionary algorithms in order to solve and optimize the *frozen lakes problem*. The goal will be to find the shortest path that leads to the goal.

We will explore how does different crossovers and mutations influence the solution, how does the probability of each also change the solution and how we can balance a fitness to have a good exploration vs exploitation ratio.

2 Representation

When thinking about the representation, we thought about two that looked promising: *The string of actions* and the *Map of action*.

2.1 The string of actions

This representation consists of a sequence of actions, numbers of 0 to 3, that is read from right to left.

Genotype:[1,3,2,0,1,1,2,2,1,2]

Initialization For the initialization of the genotypes we create lists that have a size that is the interval of the minimum steps possible to reach the goal if there were no obstacles, that is given by the square root of the map size, and twice that amount. Then we fill the list with random numbers from 0 to 3, the actions.

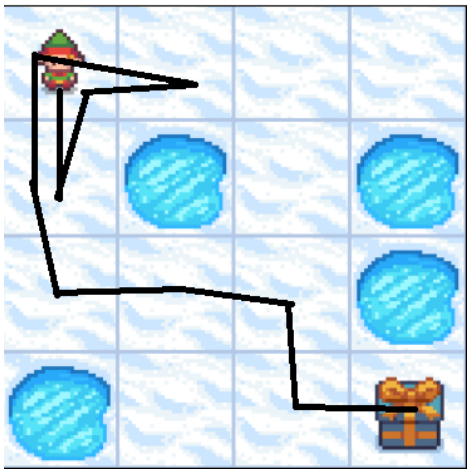

Fig. 1. String of actions

2.2 The map of actions

This representation consists of a matrix that has the same size as the map and in every cell we save the action to be performed on that cell.

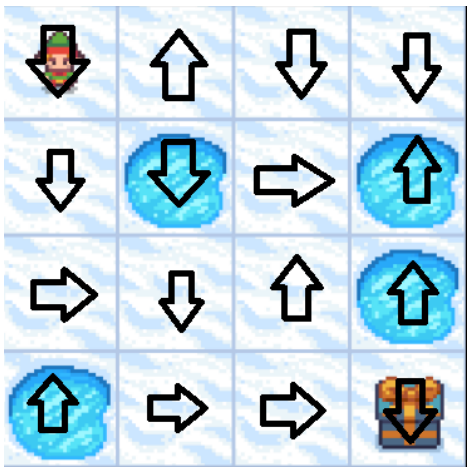

Fig. 2. Map of actions

Initialization For the initialization of the genotypes we create lists that have the same size as the map and then we fill it with random actions from 0 to 3.

After that we check the corners and the borders and make sure that the action that is there does not lead to a stuck position, for example if the top left corner had the left action assigned to it, the solution would be stuck in the top left corner until the end of the run.

2.3 Mapping

Both of the representations above will be mapped into the same phenotype, that is obtained during the simulation of the run. The result is a data structure that saves a list of the tiles that the simulation used, an integer with the number of steps used and a boolean that says if the goal was reached.

3 The evolutionary Algorithm

In this section we will explain all the elements that we used on our solution, we will look on the different crossover and mutation methods that were implemented, how does the fitness designed works, and how we perform the selection of the next generation.

3.1 Selection method

The selection is part of the algorithm that takes the elements from one population and uses them to create the population of the next generation. The selection consists of parent selection and elitism.

Parent Selection We didn´t tested nor did we explore the impact of the selection method, still the method used is *tournament selection* with a size of tournament equal to 5% of the population.

Elitism We implemented elitism in order to select the best individuals from the population and transfer them to the next one, making sure that they don't suffer any mutation.

3.2 Crossover Operations

The crossover operations are how the parents will give their genetic information to the child. In our implementations of crossover we always use two parents and we create two child's. We also leave the possibility of parent1 and parent2 being the same individual.

Uniform Crossover The crossover was only used in the mapping representation because in the string of actions one if we did this crossover we would be destroying the sequences of actions that are leading us to the goal.

One Point Crossover This crossover was used in both representations but it makes sense to use it specially in the string of actions representation since it fuses two sequences of actions from the parents, keeping their information.

3.3 Mutation Operations

One action Flip This mutation consist of changing one element of both representations and replacing it by a different one.

Line/Colon Mutation This mutation was designed for the map representation and what it does is it randomly selects a whole line or colon and changes every action in it to a new one. It made sense to create it because in this representation there are to many redundant tiles and it is a way to make change faster.

Square Mutation This mutation was designed for the map representation and what it does is it randomly applies a square of a random size to the matrix and it changes all the tiles that are in the perimeter of the square to a new action, once again it made sense to create this mutation because in this representation there are to many redundant tiles and it is a way to make change faster.

Delete action This mutation was designed for the string of actions representation and what it does is removes one action from the genotype, we used it because we verified that in this representation we end up with a lot of action not being used, the goal is reached before those actions are needed, or being redundant like hitting a wall.

3.4 Fitness

The fitness that we designed evaluates the phenotype based on 5 parameters that combined allow to minimize the redundancy of the solutions, guide them towards the goal and allow them to explore different paths.

Distance from the start position This metric uses Manhattan distance and measures how far our element ended in relation to the start position.

Steps before optimum This parameter rewards the solutions that manage to perform the minimum number of steps possible to reach the goal, for example if the map is an 12*12 the minimum number of steps that to reach the goal will be 23.

Steps post optimum This parameter rewards the steps taken after the minimum number of steps possible, it purpose is to balance the exploration and the exploitation, the bigger the reward for steps post optimum, the more the solution is encouraged to explore the map.

Goal reward This parameter is the reward that the solution receives when they reach the goal, in order to optimize solutions that are the shortest the reward is inverse proportional with the number of steps taken to reach it.

Tiles repeated This parameter is meant to punish redundancy, since an optimal solution will only pass by each tile once, so when a tile is repeated the solution is punished for doing so.

4 Experiments

Here we will show the results of the different experiments we have conducted in order to obtain the best configurations, mutations crossovers and compare both representations.

Experimental environment In the following experiments all the parameters that are not mentioned were frozen, some parameters that will remain constant in all the experiences are:

- **Population size**, it will be one hundred in all experiments
- **Number of generations**, it will be 100 in all experiments
- **Tournament selection size**, it will be 3 in all experiments
- **Map used**, the map used will be the 12*12 provided in the problem statement

Data gathering When performing experience we will save the following thing about each, the fitness of the best performing individual, the fitness of the best performing population and the generation in witch the best performing individual was obtained. For each experimental set-up we will run the program 30 times. For the tests we be using an level of significance of 5% and when comparing multiple elements we will use the *bonferroni correction*.

Understand the Distribution In order to know witch tests to use later we need to understand the distribution of your data, we need to see if it follows or does not follows a normal distribution, for that we will perform the *Kolmogorov-Smirnov* test. The null hypothesis that the data distribution follows a normal distribution.

Table 1. Results of the normal distribution test.

Data	Reject of the null Hypothesis	p-value
Best fitness	Yes	0.0011
Best Population fitness	No	0.5212
Gen best individual	No	0.7540

4.1 Comparing Configurations

For this test we will be looking at different configurations that were considered for both representations a configuration consist of a group of values for the mutation probability, the crossover probability and the size of the elite.

Map of actions In the map of actions the crossover used will be the uniform crossover and the mutation will be the one action flip. The tests that we will be performing will be *Wilcoxon rank sum* tests for each pair of values, since the best fitness does not follow a normal distribution.The null hypothesis is if the values come from continuous distributions with the same median, in other words the configuration does not have an impact in the result of the best individual.

Table 2. Results of the Wilcoxon rank sum test for the map of actions configurations best fitness

Configs	mut:1, cross:0.9 elit:2	mut:0.5, cross:0.8 elit:0	mut:0.75, cross:0.8 elit:1
mut:1, cross:0.9 elit:2	NO	NO	NO
mut:0.5, cross:0.8 elit:0	NO	NO	NO
mut:0.75, cross:0.8 elit:1	NO	NO	NO

since we got a tie, we never rejected the null hypothesis, this means that all the configurations manage to get to the same best individuals, we will now compare the speed of witch the best individual is generated. For this one the test used will be a T-test.

Table 3. Results of the T-test for the map of actions configurations best fitness population

Configs	mut:1, cross:0.9 elit:2	mut:0.5, cross:0.8 elit:0	mut:0.75, cross:0.8 elit:1
mut:1, cross:0.9 elit:2	NO	YES	NO
mut:0.5, cross:0.8 elit:0	YES	NO	NO
mut:0.75, cross:0.8 elit:1	NO	NO	NO

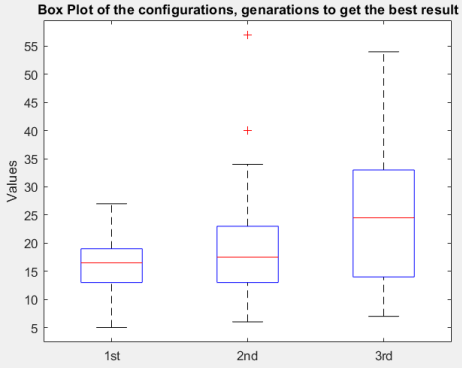

Fig. 3. Box-plot of the generations needed to hit the best individual

With this test and visual assessment of the data we were able to conclude that the ranking of the configurations is:

- **First Place**, prob mutation=1 prob crossover=0.9 and elite size=2
- **Second Place**, prob mutation=0.75, prob crossover=0.8 and elite size=1
- **third Place**, prob mutation=0.5, prob crossover=0.8 and elite size=0

From what we could observe during testing we believe that the reason why the high value of mutation is the one that performs the best is because this solution has a lot of redundancy to it and mutations on this solution respect the principle of locality, being that changes don't affect the fitness of the individual that much.

String of actions In the String of actions the crossover used will be the one point crossover and the mutation will be the one action flip mixed with the one action delete, 50% of chance for each. The tests that we will be performing will be *Wilcoxon rank sum*.The null hypothesis is if the configuration does not have an impact in the result of the best individual.

Table 4. Results of the T-test for the String of actions configurations best fitness population

Configs	mut:0.3, cross:0.8 elit:4	mut:0.1, cross:0.8 elit:0	mut:0.2, cross:0.8 elit:2
mut:0.3, cross:0.8 elit:4	NO	YES	YES
mut:0.1, cross:0.8 elit:0	YES	NO	YES
mut:0.2, cross:0.8 elit:2	YES	YES	NO

With these result we can build the following ranking for the configurations:

- **First Place**, prob mutation=0.3 prob crossover=0.8 and elite size=4

– **Second Place**, prob mutation=0.1, prob crossover=0.8 and elite size=0
– **third Place**, prob mutation=0.2, prob crossover=0.8 and elite size=2

By looking at the results, we believe that this representation is very volatile unlike the map of actions one, in the sense that changes make the quality of the solution move a lot and that the crossover implemented is not optimal for this representation since we are interrupting sequences of actions. Still we did not manage to think of a better crossover method.

4.2 Comparing Mutations

Since we only implemented one type of mutation in the string of actions representation this test will only be performed for the map of actions representation with the goal of accessing witch of your three mutations is the best one. Since there are some mutations that are more destructive than others, for example in the map 12*12, the flip action changes one tile, the line change changes 12 tiles and the square mutation changes on average 22 tiles we will have to keep that in mind when doing the tests, still we will use the same configuration for all the tests, the one found in the test above and uniform crossover. The reason for that is that we performed test that we won´t be documenting that showed that for these mutations the configuration has not a significant impact. For this one the test used will be once again the *Wilcoxon rank sum*.

Table 5. Results of the Wilcoxon rank sum for the map of actions different mutations best fitness

Mutation	Action Flip	Line Flip	Square Mutation
Action Flip	NO	NO	YES
Line Flip	NO	NO	YES
Square Mutation	YES	YES	NO

Once again we have a tie between both the Action Flip and the line flip mutation, both of them being able to reach the perfect individual in with a high consistency, in order to untie them we will use the same criteria as for the configurations and see who gets the best individuals the fastest

Table 6. Results of the T-test for the map of actions different mutations speed

Crossover	Action Flip	Line Flip
Action Flip	NO	NO
Line Flip	NO	NO

We are again in a tie, so we have reached the conclusion that even though the line flip is a lot more destructive due to the redundancy of the representation it does not seem to matter that much.

- **First Place**, Action Flip
- **First Place**, Line Flip
- **Third Place**, Square Mutation

Looking at the rank we see that the square mutation is the worst but the does the goal we tried to reach, more destructive mutations, still it seems that this is not the correct approach for the representation.

4.3 Comparing crossovers

Once again we will focus on the map of actions representation since we only have one crossover for the string of actions representation, the two crossover to be compared will be the uniform crossover vs the one-point crossover. Since both manage to get the best individuals with no significant difference we will focus on how fast they manage to get them. For this one we will use the T-test with no correction since we only have 2 crossovers.

Table 7. Results of the T-test for the map of actions different crossovers speed

Crossover	One point	Uniform
One point	NO	NO
Uniform	NO	NO

Looking at the results we see that there is no significant difference in both implementations. We believe that this comes from the nature of the representation where it is hard to have a crossover that will have a lot of impact in the representation.

4.4 Map of actions vs String of actions

Now that we found the best parameters for both configurations we will compare both of our representations. The String of actions will be using prob mutation=0.3 prob crossover=0.8 and elite size=4 and the map of actions will be using prob mutation=1 prob crossover=0.9 and elite size=2, the crossover will be the Uniform one and the mutation will be the one action flip. Both solutions reach the optimal individual, but when comparing the quality of the populations using the T-test we found the following:

Table 8. Results of the T-test for the quality of the populations Map of actions vs String of actions

Representation	Map of actions	String of actions
Map of actions	NO	YES
String of actions	YES	NO

According to the T-test we discovered that the map of actions is able to generate populations of an higher quality, this because it converges faster into the optimum solution due to the nature of the representation

Table 9. Results of the T-test for the speed Map of actions vs String of actions

Representation	Map of actions	String of actions
Map of actions	NO	NO
String of actions	NO	NO

Still in terms of speed we end in a tie, meaning that both reach an optimal solution is similar speeds.

5 Conclusion

This project successfully addressed the Frozen Lakes problem using two distinct representations. We gained insights into parameterizing representations with varying degrees of locality by adjusting mutation probabilities and mutation types. Additionally, we learned to select appropriate crossovers and mutations for different representations and to effectively assess the outcomes of evolutionary algorithms.