Robby the can cleaner - Genetic algorithm

Marko Sobak and Antonio Matosevic

Professor Hans Frisk

2015/ 10/ 10

Abstract

Evolution is a long time process of learning to adapt to some conditions in order to survive. In the last century it found its application in the field of computer science in the form of genetic algorithms, which are based on a few steps. The first one is to define simple rules, often rewards and punishments for certain forms of behaviour. We then form some individuals in a population and by calculating their fitnesses, i.e. success, imitate the natural selection. That means the best ones breed and their children, with a possibility for mutations, go to the next generation and so on. This process is repeated for a certain number of generations and it should eventually produce the best individual capable of completing the given task. In our case, a robot Robby has a task to pick up as many cans on a 10×10 grid surrounded by walls in 200 actions. We reward him for picking up a can and punish for unwanted actions. By following the presented rules, Robby will hopefully accomplish his task after a number of generations and the best individual will pick up almost all the cans without crashing into the walls or similar unwanted behaviours.

Contents

1 Introduction - A biological approach

Let us take a look at some animal species, for instance an arctic fox. If we move a fox from middle Europe up to the North Pole, it wouldn't survive for a long time because it hadn't adapted to the new living conditions. However, an arctic fox has developed a number of good characteristics in order to survive in a cold and icy area. Due to its adaptation, an arctic fox now has smaller ears then a red fox to reduce the body area and keep itself warm. Also, an arctic fox has white fur to camouflage in an ice-covered area.

Figure 1: An arctic and a red fox

This long process of adaptation to different condtitions is called evolution. We can say that animals have somehow learned what is good for them and what isn't and therefore improved the good characteristics in order to survive and have offspring. Of course, evolution includes many other factors such as breeding, crossover, mutations, natural selection, etc.

So, what does evolution have in common with a can picking robot? In the last century, biology has been an inspiration to many mathematicians and computer scientists and many ideas have emerged from it. One of them is the genetic algorithm, which is basically an imitation of the real life evolution. The idea is, in our case, to set Robby up with some very simple rules, i.e. desired forms of behaviour, which he will follow and learn to neglect actions that are not good for him.

Like in the real evolution, we will breed parents by natural selection. Hence, the better characteristic they have, the higher the chance for them to breed. We will also recombinate parents' genomes and set some probability for mutations in the gene sequence.

2 Genetic algorithm

In order to understand how the genetic algorithm works, we have to introduce some simple biological terms.

2.1 General overview of the genetic algorithm

All genetic algorithms have some steps and terms in common. The first of them is **population**, which is defined as a summation of all the individuals. An **individual** is an organism made up of a genome, which is a sequence of genes. **Fitness** is a numerical value representing the success of the individual in performing the given task.

Parents are individuals from a generation who will by **recombination** of their **genomes** form a new individual for the next population called a **child**. And finally, **mutation** is a replacement of one gene with another.

Every genetic algorithm works like this:

1. Generating a population of n individuals.
2. Calcualting the average fitness for each individual.
3. Choosing parents for breeding probabilistically. Probability depends on the fitness - the larger the fitness, the higher the chance of being chosen to breed.
4. Creating a new population of n individuals by recombinating the chosen parents' genomes.
5. Possibility for a mutation on each new child.
6. Repeating this process for some number of generations, starting from step 2.

2.2 Implementation of the algorithm in our problem

Our task includes a robot Robby who lives in a 10×10 grid surrounded by walls where each cell can be either empty or contain a can. He can move only in cardinal directions (*N, E, S, W*) and can see only his adjacent cells and cell he is standing in at any given moment. Robby's possible actions are: moves in cardinal directions, staying put, picking up a can or moving in a random direction. He has 200 actions to collect as many cans as possible.

In our case there are 3000 generations of populations and 200 individuals in each population. Also, every individual's genome consists of 243 genes. It should be mentioned that our genes will have a numerical value.

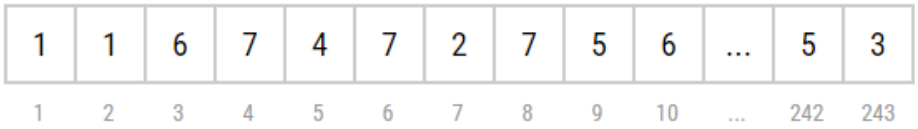

Figure 2: Genome: Genes with their indices

2.3 Genome

We know that a population is a group of 200 individuals and we know that individuals consist of a genome. However, we don't know what the genome represents in our case.

We can notice that there are 3^5 or 243 situations Robby can encounter. That is because every of five cells he can see at some point can be in any of three states (empty, can or a wall). Note that some of the situations are impossible, for instance, Robby standing in a cell that contains a wall.
Because of this we set the number of genes in an individual to 243 so that the index of each gene represents a different situation Robby can encounter. The valid question is: how does it work?

Let number 0 represent an empty cell, number 1 a cell that contains a can and number 2 a wall. If we convert every gene index, that means number of its position (1, 2, 3, ..., 243) to a number system with the base 3, we will get 243 different expressions with digits 0, 1 and 2. Since we decided to denote an empty cell as 0, one with a can as 1 and the wall as 2, we now get all the possible situation Robby can encounter. The first digit represents the cell Robby is standing in, and the others represent the adjacent cells in clockwise order, starting from the northern one. For instance, number 138 in the ternary system is written as 12010. This means that the gene at position 138 in the genome represents the situation shown in *Figure 3*.

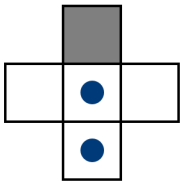

Figure 3: Situation that the gene at position 138 in the genome represents

2.4 Gene

Now that we know what each index in the genome represents, we have to explain what every gene represents. As mentioned at the beginning, Robby can take one of 7 actions in any given situation. We denoted them with numbers from 1 to 7:

1. Move north
2. Move east
3. Move south
4. Move west
5. Try to pick up a can
6. Move in a random direction
7. Stay put

We also mentioned that our genes are numerical values. Therefore, every gene, a number between 1 and 7, represents an action Robby will take in some situation. For example, if we have number 5 at position 138 in the genome, Robby will try to pick up a can. If there is a number 1, he will move north and so on.

2.5 Fitness

Now we need to develop an evaluation system of each individual's success - fitness. To do so, we calculated the fitness as a sum of rewards and punishments Robby receives for each action. We reward Robby with 20 points if he successfully picks up a can, take away 10 points if he crashes into a wall and take away 5 points if he tries to pick up a can where there isn't one.

As we already said, the parents with the best characteristics will have the highest chance to breed. This way the best individuals will have the most points, therefore the best fitness and the highest probability to breed and prolong their good genes to a new generation of children.

2.6 Recombination

In the beginning we generate 200 random individuals and let each individual try it's strategy on 100 random grids. After calculating the average fitness for each individual, we sort them by this criteria. Now we have to create a new generation of 200 individuals by breeding parents.

First we have to choose which parents will have the highest chance of breeding.
We can do it just by looking at the fitnesses. The larger the fitness, the higher the probability for

the individual to be chosen. It is basic computing of probability after adding the minimum fitness to all the others in order for all fitnesses to be non-negative:

$$p_n = \frac{f_n}{\sum_{i=1}^{200} f_i}$$

where f_n is a fitness of an individual. This method is called Roulette Wheel Selection and is often used in genetic algorithms.

After choosing two parents using the mentioned method, we recombinate their genomes. We break both genomes at some random point and recombinate them similarly to crossover in real-life genetics (*Figure 4*). Now we have two new individuals (children) which go to the next generation (population).

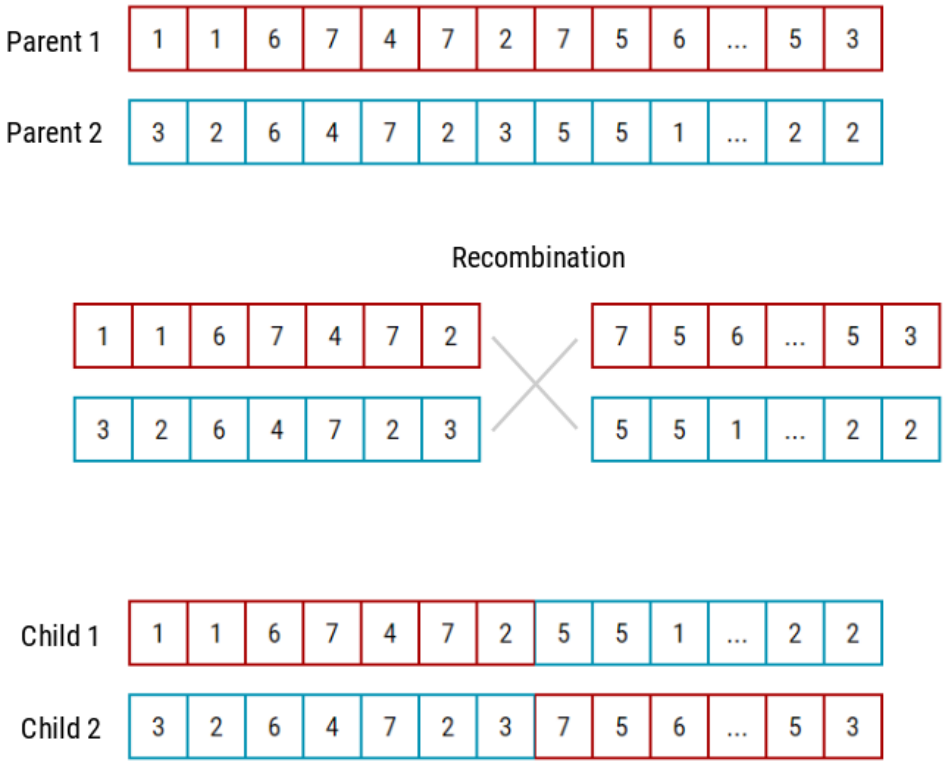

Figure 4: An example of recombination

It should be mentioned that we always send top 10 individuals from a generation to the next one to assure not to lose individuals with relatively good fitnesses due to bad mutations or recombinations.

2.7 Mutations

When we have a new population of 200 individuals, mutations may occur. We set some probability for mutations, which means they may or may not happen, just like in real life. We also set that mutations can change up to 10 genes in an individual. For example, it can change the gene on position 102 from 5 to 1. Hence, the child will no longer try to pick up a can in that situation, but move north instead.

2.8 Repeating

After this, the whole process will repeat until we reach the 3000th generation and we will hopefully get the best result at the end, i.e. the individual with fitness very close to the maximum, which means Robby picks up almost all cans and doesn't crash into the walls nor tries to pick up a can where one doesn't exist.

3 Analysis of Robby's evolution

Since now we know the algorithm for Robby's evolution, it is only left to set the number of the cans and the probability for mutations. We fixed the number of cans in the grid to 50 and changed the probability of mutations to see which one is optimal.
The probability of 25% gave us the best results so we chose that one and let evolution do the job. Here is the graph of the best fitnesses throughout evolution, that is the average fitness of the best individual in each of the 3000 generations:

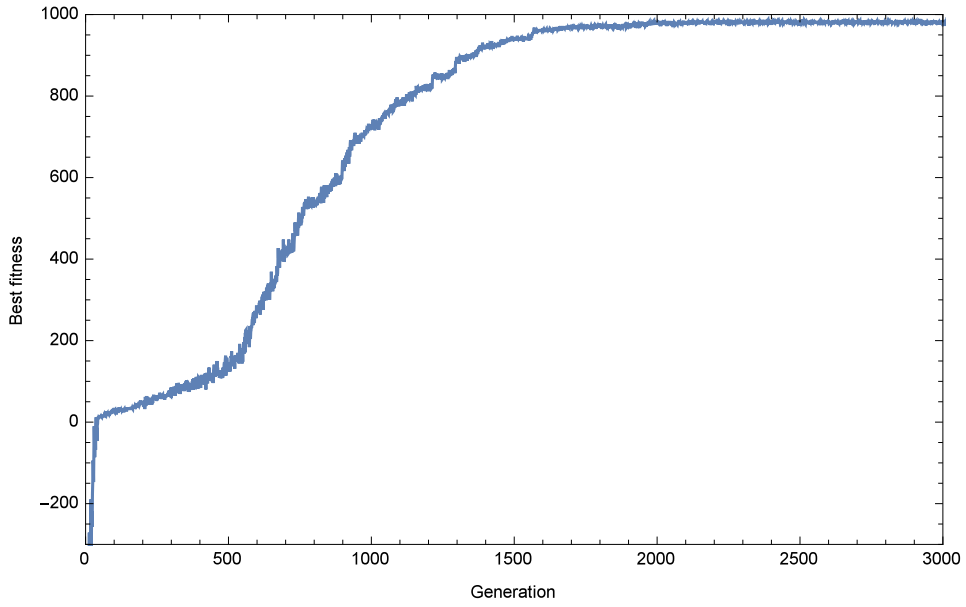

Figure 5: Growth of fitnesses during the evolution

We can see from the *Figure 5* that the average fitness of the best individual gets larger in each generation. More precisely, there is a big jump in the first few generations and an almost logarithmic growth afterwards. We can see that the graph converges around the fitness value of 985 after the 2000th generation. It should be mentioned that 1000 is the best possible fitness. That is because the reward for picking up a can is 20 points. Hence, if Robby picks up all 50 cans without punishments, his fitness will be 1000.

So we can see that evolution has succedeed in its goal, which is to develop an individual with almost perfect strategy. The question is what has Robby learned to make him so successful. Are there some bad situations he can get in?

3.1 Pros and cons of the evolutionary strategy

After all 3000 generations, we chose the best strategy (individual) and watched Robby perform on a random grid using that strategy. We noticed some interesting things evolution has developed that are not so intuitive nor logical at first.

3.1.1 Leaving cans behind

The first thing we noticed is that Robby doesn't always pick up a can if he sees one or if he is in a cell with a can. This may seem bad at the moment, but overall Robby does something quite smart.

For example, Robby can get in the situation like below and by the strategy go down:

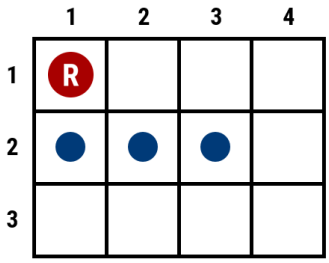

Figure 6: Robby in cell (1, 1)

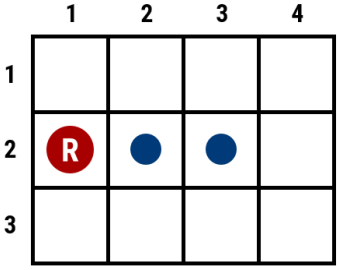

Figure 7: Robby in cell (1, 2)

What Robby does next is quite interesting. Instead of picking up a can at his current position (1, 2), he will go right. Again, he will be in a cell with a can (2, 2), but he won't pick it up. Robby will proceed to the next cell to the right (3, 2) and notice there are no more new cans to the right.

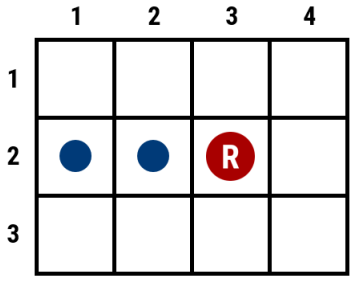

Figure 8: Robby in cell (3, 2)

When he concludes that, he will pick up the can in his current cell and go back following the same path he took to come there. However, this time, Robby will pick up all the cans he encounters and end up at position (1, 2) again.

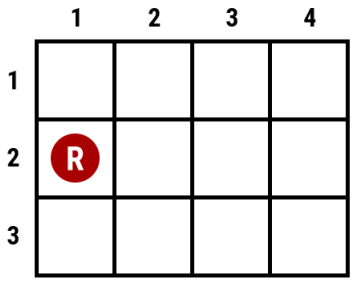

Figure 9: Robby in cell (1, 2) again

This is very smart of him because he can only see his adjacent cells so evolution took care to always have one can he has seen before so he can go back for it. Something like this wouldn't be a human's first choice, but it seems to work very well.

It should be mentioned that Robby doesn't always leave cans behind and goes back for them. If he did so, he could get into an infinite loop in a situation like one shown in *Figure 10*.

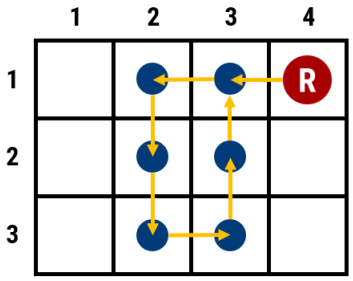

Figure 10: Infinite loop by following the cans

3.1.2 Are bad genes really that bad?

The next thing we noticed is that some of Robby's genes are equal to number 7, which means he will stay put when he gets in that situation. That also means he will stay on that cell until he takes 200 actions and the fitness in that case won't be very good. But why are those genes still there, even in some of the best individuals?

The answer is that evolution took care that Robby never gets in that situation so he doesn't need to change that gene. His strategy leads him through some path so he never gets in some situations. Hence, we can say those genes are *dead* and don't affect his fitness in any way.

3.1.3 Infinite loops

Of course, not all individuals are perfect, which means Robby sometimes doesn't pick up all the cans or maybe he crashes into walls and similar.

The best individuals mostly never crash into the walls nor try to pick up a can where there isn't one. Needless to say that Robby never stays put, even though he may have some genes which tell him to do so, but they are irrelevant, as already mentioned.

Therefore, it must be that Robby sometimes doesn't pick up all the cans. But when can that happen? Since Robby doesn't always leave cans behind, he may not collect all of them, or just by following some others cans, never come back to ones that were previously closer to him. At some point, he can get in situation like in *Figure 11*.

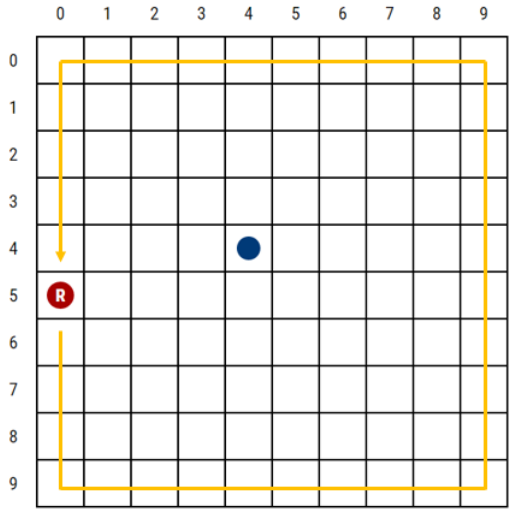

Figure 11: Infinite loop

We can notice that Robby cleaned up almost whole grid, but there is still one can left. Since there is now no way for him to know there is a can, he will just follow his strategy and circle around the grid sticking to the edge. Therefore, he will never pick it up and that will of course lower his fitness by 20. So this is an example of the situation which evolution cannot solve.

3.2 Comparison with a human made strategy

Since evolution has developed all kinds of behaviour that aren't obvious at first, we wanted to test a human made strategy with the best strategy we got from evolution.
In order to do so, we manually chose what should Robby do in all 243 situations. Of course, almost third of them are impossible so we can neglect them. By following common sense, we told Robby to pick up a can if he sees one, not to stay put and so on. We then tested both strategies on 250 random grids. And the results were following:

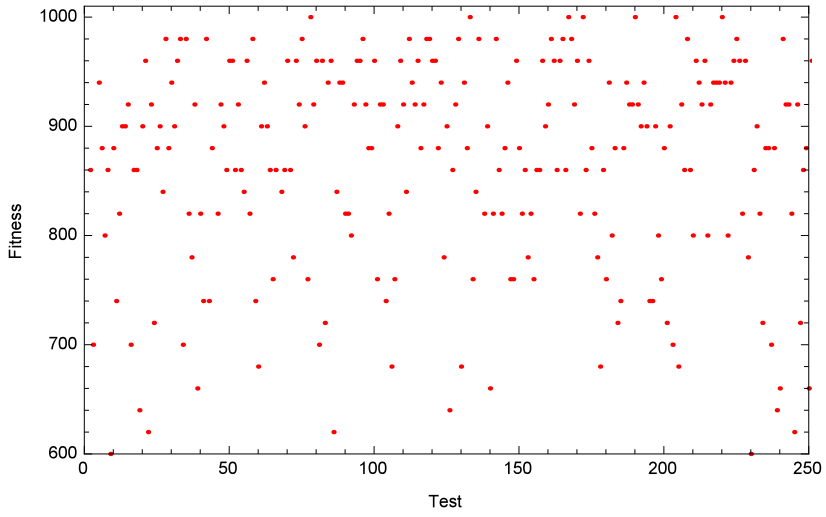

Figure 12: Human strategy

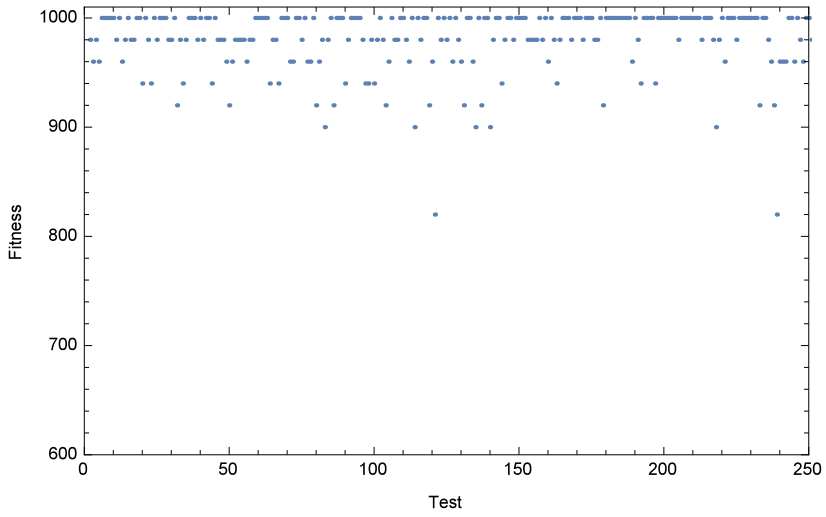

Figure 13: Evolutionary strategy

As we can see from the graphs above, the fitnesses in the human made strategy are more scattered than the fitnesses of evolutionary strategy, which are mostly around 1000. The average fitness of the evolutionary strategy is around 985 and of the human made strategy around 850. This indicates that strategy developed by genetic algorithm is in overall better than the one we made. Once again, we can see the power of the evolutionary algorithms to solve problems in a different, often not the obvious way, and beat the human intelligence.

4 Conclusion and improvements

We can conclude that the genetic algorithm and evolution work very well in the case of Robby, the can picking robot. After some number of generations he always has an individual whose fitness is quite close to the maximum one. We also noticed some interesting forms of behaviours which evolution has developed that don't seem logical at first, but prove to work even better than the human made strategies. That is why this is an example of a, so called, artificial intelligence.
However, there is still space for improvements. For example, we can calculate fitnesses and choose the best individuals in various ways. Maybe we can propose an exponential function instead the Roullete Wheel Selection so the best ones have even higher chance of being chosen. Also, we can recombinate individuals in a different way, by using uniform crossover, which includes breaking a genome in more than one point and therefore increase diversity. Some advanced genetic algorithms also use population modifiers such as incest preventing and much more, but those are just optional optimizations.