14.09.2018r.

Biologically Inspired Artificial Intelligence

Prisoner's Dilemma

Marek Skrzypecki
INF. Sem. 6, GKiO2

1. Prisoner's Dilemma

The Prisoner's Dilemma, can be formulated as follows: Two individuals (call them Mr. X and Mr. Y) are arrested for committing a crime together and are held in separate cells, with no communication possible between them. Mr. X is offered the following deal: If he confesses and agrees to testify against Mr. Y, he will receive a suspended sentence with probation, and Mr. Y will be put away for 5 years. However, if at the same time Mr. Y confesses and agrees to testify against Mr. X, his testimony will be discredited, and each will receive 4 years for pleading guilty. Mr. X is told that Mr. Y is being offered precisely the same deal. Both Mr. X and Mr. Y know that if neither testifies against the other they can be convicted only on a lesser charge for which they will each get 2 years in jail. Should Mr. X "defect" against Mr. Y and hope for the suspended sentence, risking, a 4- year sentence if Mr. Y defects? Or should he "cooperate" with Mr. Y (even though they cannot communicate), in the hope that he will also cooperate so each will get only 2 years, thereby risking a defection by Mr. Y that will send him away for 5 years? The game can be described more abstractly. Each player independently decides which move to make— i.e., whether to cooperate or defect. A "game" consists of each player's making a decision (a "move"). The possible results of a single game are summarized in a payoff matrix. Here the goal is to get as many points (as opposed to as few years in prison) as possible.

The payoff matrix:

		Player 2	
	Decision	Cooperate	Defect
Player 1	Cooperate	R=3 R=3	S=0 T=5
	Defect	T=5 S=0	P=1 P=1

What is the best strategy to use in order to maximize one's own payoff? If you suspect that your opponent is going to cooperate, then you should surely defect. If you suspect that your opponent is going to defect, then you should defect too. No matter what the other player does, it is always better to defect.

The dilemma is that if both players defect each gets a worse score than if they cooperate. If the game is iterated (that is, if the two players play several games in a row), both players always defecting will lead to a much lower total payoff than the players would get if they cooperated.

2. The best strategy

In the Prisoner's Dilemma the dominant mutual defection strategy relies on the fact that it is a one-shot game, with no future. But in the Iterated Prisoner's Dilemma (IPD) the two players may play each other again; this allows the players to develop strategies based on previous game interactions. Therefore a player‘s move now may affect how his/her opponent behaves in the future and thus affect the player‘s future payoffs. This removes the single dominant strategy of mutual defection as players use more complex strategies dependant on game histories in order to maximize the payoffs they receive. In fact, under the correct circumstances mutual cooperation can emerge. The length of the IPD (i.e. the number of repetitions of the Prisoner's Dilemma played) must not be known to either player, if it was the last iteration would become a one-shot play of the Prisoner's Dilemma and as the players know they would not play each other again, both players would defect. Thus the second to last game would be a one-shot game (not influencing any future) and incur mutual defection, and so on till all games are one-shot plays of the Prisoner's Dilemma.

Summarizing:

- The winner is the player with highest score in the end, but
- This is a Non zero Sum Game which means that both players can simultaneously win or lose.
- The number of mover should not be known to the players.
- There is no universal best strategy as the optimal strategy depends upon the opponent.

3. Genetic Algorithm

The problem of the project is to model the IPD described above and devise strategies to play it. The fundamental Prisoner's Dilemma will be used without alteration. This assumes a player may interact with many others but is assumed to be interacting with them one at a time. The players will have a memory of the previous three games only (memory-3 IPD).

A simple genetic algorithm works on the basis of the following steps:

Step 1. Start with a randomly generated population of n l-bit chromosomes (Candidate solution to a problem).

Step 2. Calculate the fitness f(x) of each chromosome x in the population.

Step 3. Repeat the following steps until n offspring have been created:

- Select a pair of parent chromosomes from the current population, the probability of selection being an increasing function of fitness. Selection is done ―with replacement‖ meaning that, the same chromosome can be selected more than once to become a parent.

- With probability Pc (the ―crossover probability‖), crossover the pair at a randomly chosen point to form two offspring. If no crossover takes place, form two offspring that are exact copies of their respective parents.

- Mutate the two offspring at each locus with probability Pm (the mutation probability or mutation rate), and place the resulting chromosomes in the new population.

Step 4. Replace the current population with the new population.

Step 5. Go to step 2.

3.1. Chromosome

For the purpose of the project the chromosome should be reporesented as 70-letter long string e.g. CCDDDCDDDCCCCCDCDDCCDCCCCDDDCDD..., where 64 letters encode a strategy (every letter is a decision for a different possibility, based on three previous moves) and 6 letters encode three hypothetical previous moves used by the strategy to decide how to move in the first actual game.
Since each locus in the string has two possible alleles (C and D), the number of possible strategies is 2^{70}.

3.2. Fitness Function

The next problem faced when designing a Genetic Algorithm is how to evaluate the success of each candidate solution. The Prisoner's Dilemma provides a natural means of evaluating the success, or fitness, of each solution – the game payoffs. We can state that the strategy, which earns the highest payoff score according to the rules of the Iterated Prisoner’s Dilemma, is the fittest, while the lowest scoring strategy is the weakest.

However these ‘raw’ fitness values present some problems. The initial populations are likely to have a small number of very high scoring individuals in a population that will take it over rapidly and cause the population to converge on one strategy.

3.3. Fitness Scaling

It is useful to scale the ‘raw’ fitness scores to help avoid the above situations. This algorithm uses linear scaling that produces a linear relationship between raw fitness – f and scaled fitness f’ as follows:

$$\mathbf{f' = af + b,}$$

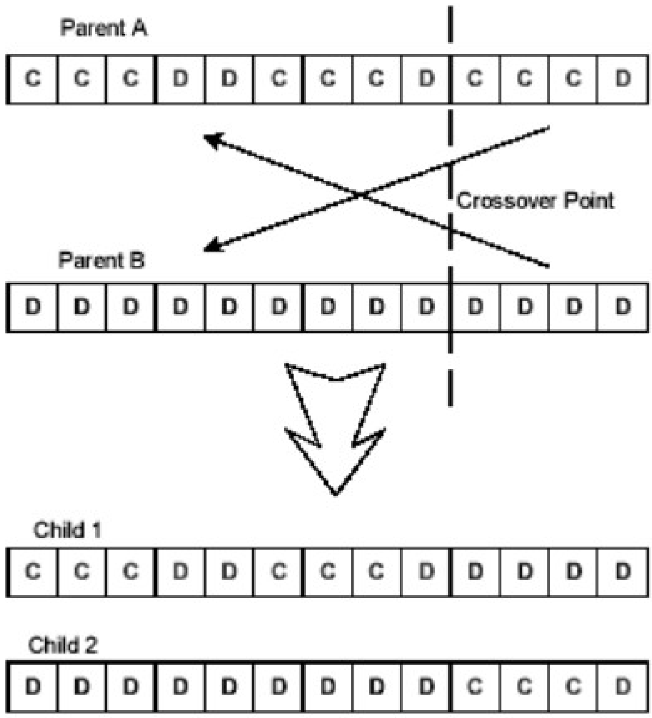
Coefficients a and b may be calculated as follows:

$$a = \frac{f_{avg}}{f_{avg} - f_{min}} \qquad b = -f_{min} \cdot \frac{f_{avg}}{f_{avg} - f_{min}}$$

Where **c** is the number of times the fittest individual should be allowed to reproduce. A value of 2 was found to produce accurate scaling in this method.

3.4. Crossover

Crossover is an artificial implementation of the exchange of genetic information that occurs in real-life reproduction. This algorithm, breaking both the parent chromosomes at the same randomly chosen point and then rejoining the parts, can implement it.



This crossover action, when applied to strategies selected proportional to their fitness, constructs new ideas from high scoring building blocks.

3.5. Mutation

Mutation will have a small possibility of occurring (0.1%) and will consist of a bit copied between the parent and the child being flipped (One letter in the chromosome string changed from C to D or D to C). These mutations provide a means of exploration to the search.

3.6. Replacement

The genetic algorithm is run across the population until it has produced enough children to build a new generation. The children then replace all of the original population.

4. Used Technology

Whole project, including genetic algorithm has been implemented using C++ programming language. Only basic libraries were used.

5. Program's Operation

Program consisits of a console application. After starting user is asked to provide the number of cycles and individuals in the population.

```
Provide number of cycles (20-200):50

Provide number of individuals in population (20-100):50_
```

After both number are provided program creates a starting, randomly generated population and simulates it's evolution. Whole population in every cycle is listed in the console along with the individuals raw and fitness scores (sorted by fitness score).

Starting population listed in console:

```
E:\Foldery\Studia\BIAI\PrisonersDilemma\x64\Release\PrisonersDilemma.exe

Provide number of cycles (20-200):50

Provide number of individuals in population (20-100):40


START


Cycle 1
Population:
DDCCDC  DCCCCCCDDCDCDDDDCDDCCDCCDDCCDCCDCCDDCDDDDCDCDCDCDDDDDCCCCCCCDDCC        6184    6548
DDCCDC  DCCCCCCDDCDCDDDDCDDCCDCCDDCCDCCDCCDDCDDDDCDCDCDCDDDDDCCCCCCCDDCC        6184    6548
DDCCDC  DCCCCCCDDCDCDDDDCDDCCDCCDDCCDCCDCCDDCDDDDCDCDCDCDDDDDCCCCCCCDDCC        6184    6548
DDCDCD  DCDDDDDCDCDDDCCDCCDCDDCDDDDCCDCDDCDCDCCDDDCCDDCCDCDDDDCCCCCCDCCC        6054    6288
DDCDCD  DCDDDDDCDCDDDCCDCCDCDDCDDDDCCDCDDCDCDCCDDDCCDDCCDCDDDDCCCCCCDCCC        6054    6288
CCDCDC  DCDDDDDDCCDCDCCDCCDDCDCCDDCDCDCDCDDDCCCCDDDDDDCCCCDDDDCCCDDCCCCCDC      5432    5044
CCDCDC  DCDDDDDDCCDCDCCDCCDDCDCCDDCDCDCDCDDDCCCCDDDDDDCCCCDDDDCCCDDCCCCCDC      5432    5044
DCCCDD  CCDDDCCDCDDCCDCDCCCDDDDDCDDDCCDDDDCDDDDCDCDCDDCDDDCCDDDCCCCCCCDD        5416    5012
DCCCDD  CCDDDCCDCDDCCDCDCCCDDDDDCDDDCCDDDDCDDDDCDCDCDDCDDDCCDDDCCCCCCCDD        5416    5012
DCCCDD  CCDDDCCDCDDCCDCDCCCDDDDDCDDDCCDDDDCDDDDCDCDCDDCDDDCCDDDCCCCCCCDD        5416    5012
CDCCDD  CCCCDDDDCCDCDDDCDCDDCDCCDDCDCDCCCCCCCDCDDCDDDDDCDCDDCDCCCCCDCDCC        5368    4916
CDCCDD  CCCCDDDDCCDCDDDCDCDDCDCCDDCDCDCCCCCCCDCDDCDDDDDCDCDDCDCCCCCDCDCC        5368    4916
CDCCDD  CCCCDDDDCCDCDDDCDCDDCDCCDDCDCDCCCCCCCDCDDCDDDDDCDCDDCDCCCCCDCDCC        5368    4916
DDDCDD  CCCCCDDDDDDDCDCDDDDCDCCCDDCDDDDDCCDDDDDCDDCDDCCDDCDDDDDDDCCDCCCD        5292    4764
DDDCDD  CCCCCDDDDDDDCDCDDDDCDCCCDDCDDDDDCCDDDDDCDDCDDCCDDCDDDDDDDCCDCCCD        5292    4764
DDDCDD  CCCCCDDDDDDDCDCDDDDCDCCCDDCDDDDDCCDDDDDCDDCDDCCDDCDDDDDDDCCDCCCD        5292    4764
DCDDCC  DDDDDCDCDDDDCDCDCCCCDDCDDCCCCDDCCCCDCDCCDDCDDCDDDCDCCCDCCCCDCCDD        5284    4748
DCDDCC  DDDDDCDCDDDDCDCDCCCCDDCDDCCCCDDCCCCDCDCCDDCDDCDDDCDCCCDCCCCDCCDD        5284    4748
DCDDCC  DDDDDCDCDDDDCDCDCCCCDDCDDCCCCDDCCCCDCDCCDDCDDCDDDCDCCCDCCCCDCCDD        5284    4748
CDCDDD  DCDCDCCDDCDCDCCDCCCDDDDCDDDCCCDCCDDCDCCDDDCCCDDDCDCCDDCCDDCCCCDC        5258    4696
CDCDDD  DCDCDCCDDCDCDCCDCCCDDDDCDDDCCCDCCDDCDCCDDDCCCDDDCDCCDDCCDDCCCCDC        5258    4696
DCDDDC  DCDCDDCCDDCCDCCCDDDDCDDCCDCCCCDCDDCCCDDDDCCCCCCCDCCDCCCDDDCDCDDD        5134    4448
DCDDDC  DCDCDDCCDDCCDCCCDDDDCDDCCDCCCCDCDDCCCDDDDCCCCCCCDCCDCCCDDDCDCDDD        5134    4448
DDCDCD  CDDCDCDCCDDDDCCCCDCCDCDDCCCDDCDCDDCDDCDCDCCDCDCDDDDCDDDCDCCDCDCD        5118    4416
DDCDCD  CDDCDCDCCDDDDCCCCDCCDCDDCCCDDCDCDDCDDCDCDCCDCDCDDDDCDDDCDCCDCDCD        5118    4416
CDCCDC  CCDDDCCCCCDCCCCCDCDCDDDDDDDCCDDDDDDCDCCDCCDDCCCCDDCDDDCCCDDCCDC         5076    4332
CDCCDC  CCDDDCCCCCDCCCCCDCDCDDDDDDDCCDDDDDDCDCCDCCDDCCCCDDCDDDCCCDDCCDC         5076    4332
DCDDCC  CDDDCDCCCDDCCDCCCCDDCCDCDDDCCDDCCDDDCDCCDCDCDCCDDCDCCCDCCCDDDDCD        5056    4292
DCDDCC  CDDDCDCCCDDCCDCCCCDDCCDCDDDCCDDCCDDDCDCCDCDCDCCDDCDCCCDCCCDDDDCD        5056    4292
CDCCDD  CDCCDCCDCDDDDDCDDCCDCCCCCCCCCCCDCDDDDDCDDDCDDDDDCCCCDDCCCDCDDDC         4930    4040
CDCCDD  CDCCDCCDCDDDDDCDDCCDCCCCCCCCCCCDCDDDDDCDDDCDDDDDCCCCDDCCCDCDDDC         4930    4040
DDDDDD  DDCCDCCDCDCDCDCCDCCCDDCDDCDCDCCDCCDCCCCDDCCDCDDDDCCDDCCDDCCCDCDD        4878    3936
DDDDDD  DDCCDCCDCDCDCDCCDCCCDDCDDCDCDCCDCCDCCCCDDCCDCDDDDCCDDCCDDCCCDCDD        4878    3936
DDDDDD  DDCCDCCDCDCDCDCCDCCCDDCDDCDCDCCDCCDCCCCDDCCDCDDDDCCDDCCDDCCCDCDD        4878    3936
CDDDDC  DCCDDCCDDCDDDCCDCCDDDDCCCCDCDDCCDCDCCCCDDCCCCDDCCDDDDCCDDCDCCCDD        4600    3380
CDDDDC  DCCDDCCDDCDDDCCDCCDDDDCCCCDCDDCCDCDCCCCDDCCCCDDCCDDDDCCDDCDCCCDD        4600    3380
CDDCDC  CDDDDCDDDDDCCDCDDCCCCCDDDCDDCCDDDCCCCDCCDDDCCCDDDCCCCCDCDCDDCDCC        3918    2016
CDDCDC  CDDDDCDDDDDCCDCDDCCCCCDDDCDDCCDDDCCCCDCCDDDCCCDDDCCCCCDCDCDDCDCC        3918    2016
DCCDCD  DDCDCCCCDDDDCDCDCDCDCCCCCCDCCDCDCCDCDCDDDCDDCCDDDDCDCDCCDCCDDDDD        2910    0
```

Final population listed in console:

```
WybierzE:\Foldery\Studia\BIAI\PrisonersDilemma\x64\Release\PrisonersDilemma.exe
Cycle 50
Population:
CCDDDC  DCDDDCCDCCCDCDCDDCCDDDDDCDDDCCDDDDCDDDDCDCDCDCCDDDCDDDCCCDCCCCCC       12100   22788
DCCCCD  DCCCDCCDDCDDCCCDCCCCCDDDCDDDCCDDDDCDDDDCDCDCDCCDDDCDDDCCCCCDCCCC       6264    5280
CCCCCD  DCCCDCCDDCDCDDCDCCCCDDDDCDDDCCDDDDCDDDDCDCDCDCCDDDCDDDCCCCCDDDCC       6262    5274
DCCCCD  DCCCDCCDDCDCDDCDCCCDDDDCCDDDCCDDDDCDDDDCDCDCDCCDDDCDDDCCCCCDDDCD       6260    5268
DCCCCD  DCCCDCCDDCDCDDCDCCCDDDCCCDDDCCDDDDCCDDDCDCDCDCCDCDCCDDCDCCCDDCCC       6258    5262
DCCCCD  DCCCDCCDDCDCDDCDCCCCDDDDCDDDCCDDDDCDDDDCDCDCDCCDCDCDDDCCCCDDCCCC       6242    5214
DCCCDC  DCCCDCCCCCCDCCCCCCCDDDDCCDDCCCCDDDCDDDDCDDDCDDCDCDCCDDCCCCCDDDCD       6236    5196
DCCCDC  DCCCDCCDCCDDCCCCCCCDDDDCCDCCCCCCDDCDDDDCDDDDDDCDCDCCDDCCCCCDCCCC       6236    5196
DCCCDC  DCCCDCCCCCCDCCCCCCCDDDDCCDDDCCDDDDCDDDDCDDDCDDCDCDCCDDCCCCCDCDCC       6236    5196
DCCCDD  DCCCDCCDDCDCDDCDCCCCDDDDCDDCCCDDDDCDDDDCDDDCDCCDDDCDDDCCCDCCCCCC       6224    5160
DCCCDD  DCCCDCCDDCCDDDCCCCCDDDCCCDDCCCCDDDCDDDDCDCDCDCCDDDCDDDCCCDCDDDCC       6224    5160
DCCCDC  DCCCDCCCDCCDDCCDCCCCDDDDCDDDCCDDDDCDDDDCDCDCDCCDDDCDDDCCCCCDCCCC       6216    5136
CCDDDC  DCCDDCCDDCCDCDCDCCCDDDDDCDDDCCDDDDCDDDDCDCDCDCCDDDCCDDCDCCCDDCCC       6212    5124
CCDDDC  DCDDDCCDDCDCDDCDCCCCDDDDCDDDCCDDDDCDDDDCDDDCDCCDDDCCDDCCDCCDDDCC       6208    5112
DCCCDC  DCCCDCCDCCDCDDCDCCCDDDDCCDDDCCDDDDCDDDDCDCDCDCCDDDCCDDCCCCCDDDCC       6208    5112
DCCCDC  DCCCDCCDDCCDCCCCCCCDDDCCCDDCCCDDDDCDDDDCDCDCDCCDDDCDDDCCCCCCCCCC       6206    5106
DCCCDC  DCCCDCCDDCCDDDCCCCCDDDCCCDDCCCCDDDCDDDDCDCDCDDCDDDCDDDCCCCCDCCCC       6206    5106
DCCCCD  DCCCDCCDDCDCDDCDCCCDDDCCCDDDCCDDDDCCDDDCDCDCDCCDCDCCDDCCCCCDCCCC       6202    5094
DCCCDC  DCCCDCCDCCDDCCCCCCCDDDCCCDDCCCDDDDCDDDDCDCDCDDCDDDCCDDCCCCCDDDCC       6200    5088
DCCCDC  DCCCDCCDCCDDCCCCCCCDDDCCCDDCCCDDDDCDDDDCDCDCDCCDDDCDDDCCCCCDDCCC       6200    5088
DCCCDC  DCCCDCCDCCDDCCCCCCCDDDDCCDDDCCDDDDCDDDDCDDDCDDCDDDCCDDCCCCCDCCCC       6200    5088
DCCCDC  DCCCDCCDCCDDCCCCCCCDDDCCCDDCCCDDDDCDDDDCDDDCDDCDDDCCDDCCCCCDCCCC       6200    5088
DCCCDC  DCCCDCCDCCDDCCCCCCCDDDCCCDDCCCDDDDCDDDDCDCDCDCCDDDCDDDCCCDCCCCCC       6200    5088
CCCCCD  DCDDDCCDDCCDCDCDCCCCDDDDCDDCCCDDDDCDDDDCDDDCDCCDDDCDDDCCCDCCCCCC       6198    5082
DCCCDC  DCCCDCCDCCDDCCCCCCCDDDCCCDDDCCDDDDCDDDDCDCDCDCCDDCCCDDCDCCCDDCCC       6196    5076
DCCCDD  DCCCDCCDDCCDDDCCCCCDDDCCCDDCCCDDDDCDDDDCDCDCDCCDDDCDDDCCCDCCCCCC       6184    5040
CCDDDC  DCDDDCCDCCCDCDCDDCCDDDDDCDDDCCDDDDCDDDDCDCDCDDCDCDCDDDCCCCCDDCCC       6184    5040
DCCCCD  DDDDDCCDDCDCDDCCCCCDCDDCCDDDCCCDDDCDDDDCDCDCDCCDCDCDDDCCCCDDCCCC       6178    5022
DCCCDC  DCCCDCCDDCCDCCCCCCCDDDCCCDDCCCDDDDCDDDDCDCDCDCCDDDCDDDCCCCCDDCCC       6150    4938
DCCCDD  DCCCDCCDDCDCDDCDCCCCDDDDCDDDCCDDDDCDDDDCDCDCDCCDDCCCDDCCCCCDDDCC       6102    4794
CCDDDC  DCDDDCCDDCCDCCCDCCCCDDDDCDDDCCDDDDCDDDDCDCDCDCCDDCCCDDCDCDCDDCCC       6102    4794
CCDDDC  DCDDDCCDDCDCDDCCCCCCDDDDCDDDCCDDDDCDDDDCDCDCDCCDDDCDDDCCCDCCDDCC       6086    4746
CCDDDC  DCDDDCCDDCCDCCCDCCCCDDDDCDDDCCDDCDCDDDDCDDDCDDCDDDCCDDCCCCCDCCCC       6084    4740
CCDDDC  DCDDDCCDDCCDCCCDCCCCDDDDCDDDCCDDCDCDDDDCDDDCDDCDDDCDDDCCCCCDDDCC       6084    4740
DCCCCD  DCCCDCCDDCDCDDCCCCCCDDDDCDCDCCDDDCCDDDDCDCDCDDCDDDCCDDCDCDCDCCDC       6062    4674
DCCCCD  DDDDDCCCDCDCDDCCCCCDCDDCCDDDCCDDDDCDDCDCDCDCDCCDDDCDDCCCCCCDCCCC       6042    4614
CCDDDC  DCDDDCCDDCDCDDCDCCCCDDDDCDDDCCDDDDCDDDDCDDDCDDCDDDCCDDCCCCCDCCCC       6020    4548
CCDDDC  DCDDDCCDCCCDCDCDDCCDDDDDCDDDCCCDDDCDDDDCDCDCDCCDDDCDDDCCCDCCCCCC       6014    4530
CCDDDC  DCDDDCCDCCCDCDCDDCCDDDDDCDDDCCDDDDCDDDDCDDDCDCCDDDCDDDCCCDCCCCCC       6014    4530
DCCCDC  DCCCCCCDDCDCCDCCCCCDDDDDCDDDCCDDDDCDCDDCDDDCDCCDDDCCDDCCDCCDDDCC       4504    0
END

Press any key to continue . . .
```

The starting and final populations are also written to "Results.txt" file.

By just looking on the results listed in console we can see the differences in both populations. Staring population consists of various individuals which are quite different from each other. Their raw scores (left column) are more evenly spread beetwen the highest and lowest scoring individuals.

However in case of the final population we can see that individuals are more similar to each other, especially in specific areas. This is a result of selecting only individuals with parts of chromosomes that quarantee high scores to produce the next generations. Looking at the scores (apart from the singular highest and lowest scoring individuals) we can see whole population with really similar scores. But whats important is that the score that seems to be average in the final population is as good if not better than the score of the best individual of the starting population.

6. Conclusions

As for my thoughts and conclusions about the project I have to say that I was surprised at the first results of my work, which were actually pretty good, despite not using any ready libraries or technologies dedicated for this kind of work.

While reading about Prisoner's Dilemma I got to learn a lot of variations of this game and it's reflection in real world phenomenon like game theory, economics or political science. I also came across other similar games like Mastermind in which players are developing kind of a strategy and genetic algorithms could be applied to do that.

Working on this project helped me to understand how genetic algorithms work and see that there are many applications to them.

GitHub repository: GitHub
Google Drive: GoogleDrive