GENETIC ALGORITHMS PROJECT

Revanth Mattey

Shashank Pathrudkar

Computational Intelligence – CS/EE 5821

Submission Date: 12/06/2019

ACKNOWLEDGEMENT

We would like to thank Dr. Tim Havens for designing this activity which helped us learn about Genetic Algorithms.

We would like to thank Michigan Tech IT services to provide me MATLAB License.

We are grateful towards J. Robert Van Pelt and John and Ruanne Opie Library to make the library computers available for this project.

Lastly, we would thank my fellow classmates who asked their doubts to Dr. Havens during classroom sessions. Responses of Dr. Havens to those doubts helped us understand the concepts further.

Contents

Problem Statement 4
PROBLEM 1: Genetic Algorithm for Optimization. 6
PROBLEM 2: Genetic Algorithm for Iterated Prisoners Dilemma 11
Conclusions: 14
REFERENCES 15

PROJECT -3

Problem Statement

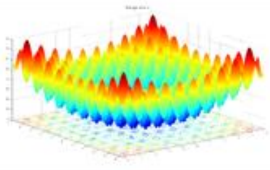
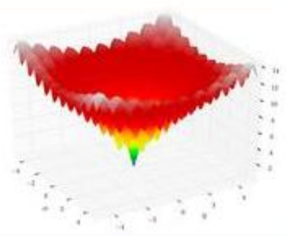
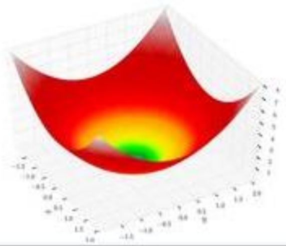
PROBLEM 1: Genetic Algorithm for Optimization.

To develop a Genetic algorithm for optimizing a fitness function, $f(x), \vec{x} \in R^d$, where the values of $\vec{x}$ are in the range $\left[-\frac{xrange}{2} : \frac{xrange}{2}\right]$.

Here, we develop and the test the code for minimizing the following three fitness functions:

1. Dejong
2. Rastrigin
3. Ackley

The functional form and the graphs of the three fitness functions are as given below:

Name	Plot	Formula
Rastrigin function		$f(\mathbf{x}) = An + \sum_{i=1}^{n} \left[x_i^2 - A\cos(2\pi x_i)\right]$ where: $A = 10$
Ackley function		$f(x,y) = -20\exp\left[-0.2\sqrt{0.5\left(x^2+y^2\right)}\right] - \exp\left[0.5\left(\cos 2\pi x + \cos 2\pi y\right)\right] + e + 20$
Sphere function		$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$

***Source**:https://en.wikipedia.org/wiki/Test_functions_for_optimization*

All the functions have a global optimum at O(Origin). Sphere function also known as Dejong.

PROBLEM 2: Genetic Algorithm for Iterated Prisoners Dilemma

The prisoner's dilemma is a game between 2 players. Imagine that two criminals, P1 and P2, have committed a crime and have been detained. Each is placed alone in an interrogation cell and given the same two choices: "testify" or "silence." Now using the information about the sentence, we develop a GA for finding out a best possible strategy given an opposition's strategy. Below given are four possible outcomes in each case.

	P2 testify	P2 silence
P1 testify	P1 = P2 = 3 yrs	P1 goes free, P2 = 5 yrs
P1 silence	P1 = 5 yrs, P2 goes free	P1 = P2 = 1 yr

PROBLEM 1: Genetic Algorithm for Optimization.

Q1.

Give the best fitness function value and associated $\vec{x}$ over 10 runs for each of the three benchmark functions (use 3 or more dimensions in at least two of the benchmark functions). Include results of runtime (hint: use tic and toc). Present your results in a table.

Dejong

Run Number	xrange					fxbest	xbest					Time Elapsed(secs)
1	242	259	916	448	529	0.0001	0	-0.01	0	0	0	90.613
2	300	917	726	890	688	0	0	0	0	0	0	84.572
3	739	977	215	57	441	0	0	0	0	0	0	84.339
4	553	905	162	190	835	0.0001	0	0	0.01	0	0	101.957
5	554	310	143	293	610	0.0001	0	0	-0.01	0	0	90.480
6	590	166	981	370	843	0.0001	0	0	-0.01	0	0	139.184
7	683	739	631	214	703	0.0001	-0.01	0	0	0	0	112.752
8	363	172	171	617	59	0.0001	0.01	0	0	0	0	95.318
9	931	609	418	968	549	0.0001	0	0.01	0	0	0	54.567
10	881	915	903	236	354	0	0	0	0	0	0	56.832

Rastrigin

Run Number	xrange					fxbest	xbest					Time Elapsed(secs)
1	618	412	504	613	926	0	0	0	0	0	0	50.144
2	648	589	965	549	214	0.0198	0	-0.01	0	0	0	45.188
3	448	447	179	702	198	0	0	0	0	0	0	49.339
4	274	583	807	422	737	0	0	0	0	0	0	50.008
5	399	617	36	614	732	0	0	0	0	0	0	44.748
6	699	50	630	909	796	0	0	0	0	0	0	46.770
7	131	337	912	87	891	0	0	0	0	0	0	44.775
8	258	29	179	976	326	0	0	0	0	0	0	42.104
9	211	202	67	756	808	0	0	0	0	0	0	42.672
10	693	417	133	597	946	0	0	0	0	0	0	49.595

Ackley

Run Number	xrange					fxbest	xbest					Time Elapsed(secs)
1	943	330	119	509	203	8.88E-16	0	0	0	0	0	42.011
2	821	409	304	511	903	8.88E-16	0	0	0	0	0	60.378
3	427	230	931	560	118	8.88E-16	0	0	0	0	0	69.335
4	224	662	142	708	943	8.88E-16	0	0	0	0	0	51.732
5	243	978	13	134	277	8.88E-16	0	0	0	0	0	47.420
6	260	123	436	918	439	8.88E-16	0	0	0	0	0	49.868
7	779	145	17	912	815	8.88E-16	0	0	0	0	0	61.883
8	647	950	222	651	472	8.88E-16	0	0	0	0	0	62.213
9	487	874	545	988	332	8.88E-16	0	0	0	0	0	66.452
10	809	404	204	295	569	8.88E-16	0	0	0	0	0	64.928

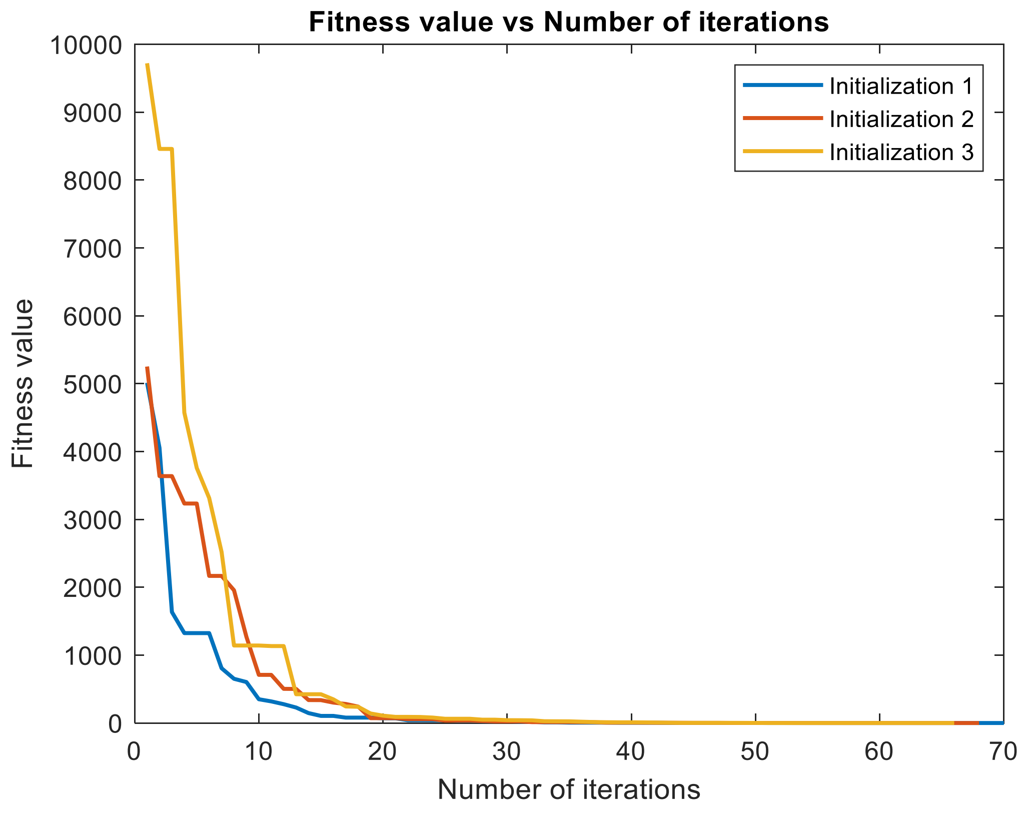
Remarks:

There may be some inconsistencies in the time measured as this would be based on the performance of the device on which we ran the code. But in general we can observe that the runtime stays at an average of around 60 secs. This can be considered good given that the xrange varies a lot in each and every instantiation. When we see the value of xbest obtained **Ackley** performs as it has a significantly comprehensible global minima. So it easily optimizes compared to Rastrigin and Dejong.

Q2.

Produce a plot of the best $f(\vec{x})$ value at each iteration vs iteration number for the Rastrigin benchmark function. Show at least 3 different curves on this plot (for three different initializations).

Rastrigin												
Initialization	**xrange**					**fxbest**	**xbest**					**Time Elapsed(secs)**
1	287	324	896	527	562	0	0	0	0	0	0	46.141
2	974	852	831	194	994	0	0	0	0	0	0	43.733
3	671	897	785	752	300	0.0198	0	0	0.01	0	0	43.057



Remarks:

It can be observed that there is an overall decreasing trend in the best fitness values. And also, the GA converges after 30-40 iterations in each and every initialization.

Q3.

Describe your GA so that someone could reproduce your results. E.g., what parameter values did you use (termination criteria, population size, crossover/mutation probabilities, etc.)? How did you encode your genes? How did you perform crossover and mutation? Did you use elitism? I should be able to use your text to reproduce your code and results.

First, we started by laying down a plan on what strategies should be implemented. Below given is the chronological order of the steps implemented:

1. Generating initial population in the specified range of $\left[-\frac{xrange}{2} : \frac{xrange}{2}\right]$. We have initially generated 2000 samples in our initial population.
2. Computing the cumulative norm of the fitness and perform reproduction from the initial samples. From the 2000 initial population now we reproduce 1000 parents by using **Baker's Roulette wheel method**.
3. After generating the initial samples we choose the probability of crossover and mutation.
   Prob_Crossover = 0.75;
   Prob_Mutation = 0.02;
4. Now we implement 2 point crossover after pairing the strings randomly.
5. Once crossover is performed, we then move on to mutation. We initially determine the number of mutations by multiplying the probability of mutation with the total length of all the strings combined. Once we get the number of mutations then we generate random indexes for the location of the mutation.
6. After mutation we then decode the binary strings to numeric arrays and then evaluate the fitness of the mutated and crossed over parents.
7. Once the fitness is evaluated then by using the **Elitism** strategy we retain the two best individuals.
8. An additional strategy which we used is we defined an elite percentage of 30 and carry forward the best performing(based on fitness values). Now these members combined with the new reproduced members from the bakers method are combined together and the same iterative procedure is followed until the stopping criteria is met.
9. **Termination Criteria**: We define the termination criteria as follows:
   If the difference of fitness in 5 continuous generations > 0.0001 || ($50 <$ Generations < 500)

Strategy used for Encoding and Decoding:
For Encoding in each generation to make the code efficient, we adopt a dynamic scheme. That is in each iteration the max value in the array is taken and the number of bits required are determined. So in this way we can allow a dynamic encoding scheme by increasing the code performance and reducing the run time. For representing positive and negative signs we use an extra bit at the start of the bit string with "**1**" as negative and "**0**" as positive. For floating points we consider a maximum up to 2 digits after the decimal point. So we use 7 bits to encode the float values.
Also after generating the binary strings now by using these strings to generate **gray coded binary strings** using the XOR logic.
While Decoding the same logic as mentioned above is used but just in a reverse fashion.

Q4.

Describe the process you went through to develop the final instantiation of your GA. E.g., did you try different parameters, GA types, etc.? How did you decide on your final parameter values?

During the process of developing the final instantiation of the GA, we went through a lot of brain storming. Initially for reproduction the samples were generated randomly. Also we didn't use the strategy (mentioned in point **8** of the earlier question) of retaining the best performing parents at each and every iteration and performing operations like mutation and crossover.

But later we improved the performance by using techniques like Elitism, Bakers Method and dynamic encoding which helped us to increase the code performance as well as increase the accuracy of the final result. Initially we had results in the range of 2% of the optima, but after implementing all the described strategies we achieved results in the range of sub 0.5% for any xrange. We had also tried changing the number of **individuals reproduced** in each generation and **1000 individuals** proved to be optimal with regard to accuracy and computational time.

The final parameter values like probability of crossover and probability of mutation are selected based on the accuracy of results we achieved. The values of 0.75 and 0.02 for crossover and mutation respectively proved to be the best values.

PROBLEM 2: Genetic Algorithm for Iterated Prisoners Dilemma

Q1.

Based on your knowledge (or research) of this game, what is the best strategy for the two prisoners?

Robert Axelrod organized tournaments on iterated prisoner's dilemma where academicians from whole world participated. He discovered that greedy or selfish strategies always perform poorly as compared to more **altruistic strategies**. In his tournament the winner was **tit for tat** strategy which is the best strategy for the game. Tit for tat strategy cooperates on the first iteration of the game and then copies opponent's previous iteration move.

Q2.

Assuming that the prisoners have no knowledge of the other prisoner's past behavior (this is the first time they have been caught), how many bits do you need to encode a GA to solve this problem? (hint: the answer is 1) Why is it 1?

The prisoners have only 1 decision to make with only 2 options available. **Binary decision** can be encoded in **one bit**. The bit would carry a value '1' or '0' according to the decision made.

Q3.

If you were going to design a GA to solve this problem, how would you express the fitness function? (Note: you don't have to build the GA for this question... just think about how you would judge the fitness of each member of the population)

Decisions made by the members at every crime result in either 0,1,3,5 years of imprisonment. So, for *n* crimes a member of population can get a maximum of *n**5 years of imprisonment. The fitness function can therefore be ***n*5 – years of imprisonment***. Note that we must **maximize this function** in order to minimize years of imprisonment. *n*5* would give a value of 50 for 10 crimes. Years of imprisonment can be calculated based on the sentence player receives in 10 crimes.

Q4.

Will these prisoners act the same as they do for the memory-0 IPD? Why or why not? (hint: check out Axelrod's 1970s book, The Evolution of Cooperation, for more info on this. There are also many online references that discuss IPD)

Memory-0 IPD does not give players the information about their or opponents past decisions. In memory-2 IPD prisoners know about their opponent's past 2 decisions, so they would always tend to decide accordingly. **It won't be same as memory-0 IPD**. For example, for an always non cooperative partner, a player would also decide to have non cooperative strategy. History of decisions help player in deciding their strategy according to greedy or cooperative nature of their opponent.

Outline of GA Strategy for Memory 2 IPD (Discussed in further questions)

Q5.

What is your fitness function: How do you judge a specific strategy?

Our fitness function is ***number of crimes*5 – years of imprisonment*.** We maximize this function. *n*5* would be the maximum imprisonment for n crimes. Years of imprisonment is calculated according to the decision made by player and their opponent in a spree of 10 crimes.

To judge a specific strategy, we assume that the **criminals commit 10 crimes** (We assume that they commit 10 crimes as it would result in an average imprisonment of 25-30 years for both. Realistically, 25 years of imprisonment is significant amount in human life). According to the player strategy and opponent strategy we **add up their years of imprisonment** and check if the player strategy we devised is yielding in lesser years of imprisonment for the player than the opponent.

To devise a strategy for our player we will have to have an opponent strategy. The GA won't always generate same player strategy. It depends a lot on opponent's strategy, as years of imprisonment (which is fitness) not only depends on players decision but also depends on opponent's decision.

Q6.

How do you encode your genes? (There is a lot of information online about GAs and IPD.)

The player would always have a binary decision to make. Binary decisions can be encoded in bits. We considered ourselves as consultants for criminals. Criminals come to us and ask for strategy. We want to make sure that the strategy we provide should help the criminal to take decision for any *n*th crime (*n* starting from1). The player has a knowledge of previous 2 decisions that they and their opponent made (Memory-2). To achieve this objective, we need to give them strategy for 1st crime, 2nd crime and nth crime (n=3, 4, ….). Strategy for 1st crime is encoded in 1 bit (a simple binary decision whether to testify or stay silent). For 2nd crime we need 4 bits for 4 different possible cases. From 3rd crime onward player would have memory of previous 2 crimes, which would have 16 different cases. Therefore, we need another 16 bits for this. A total of 1+4+16=**21 bits** are used by us to store the strategy. Details of the encoding and strategy are in the table below.

Note that 2 bits from 2nd crime 4 bits would be redundant based on the players decision of their 1st crime.

	1st crime	2nd crime				nth crime															
Player's (n-2)th decision	-	-	-	-	-	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
Player's (n-1)th decision	-	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
Opponent's (n-2)th decision	-	-	-	-	-	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
Opponent's (n-1)th decision	-	1	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Player's decision	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**

0: Testify

1: Silence

Q7.

On how many turns do you test a strategy in order to calculate your fitness? Why?

To judge a specific strategy, we assume that the **criminals commit 10 crimes** (We assume that they commit 10 crimes as it would result in an average imprisonment of 25-30 years for both. Realistically, 25 years of imprisonment is significant amount in human life).

Q8.

Take the strategy your GA finds and test it against prisoners that i) always testify, ii) always stay silent, iii) randomly choose, and iv) the tit-for-tat strategy for 10 turns. How many years did your GA prisoner get as compared to the other strategies? Comment on the results.

	Years of Imprisonment (10 crimes)	
Opponent Strategy	**Player**	**Opponent**
Always testify	30	30
Always Silence	5	30
Random (Averaged for 100 opponents)	18.75	31.45
Tit for tat	24	29

The strategy generated by GA algorithm helps the player to get less years of imprisonment for 3 out of 4 opponent strategies. For always testifying opponent, player strategy gives the player same years of imprisonment. Years of imprisonment is almost same in case of Tit for tat opponent. The results change according to the player strategy. In some cases, we also observed more imprisonment for player than always testifying opponent. Player strategy depend a lot on what opponent strategy you use while generating player strategy.

Conclusions:

- Part 1 of this project made us learn use of genetic algorithms to do optimization and minimize/maximize a given function.
- Genetic Algorithm proves to be helpful in finding global minimum for functions like rastrigin which have many local minima.
- Part 2 made us apply genetic algorithm to a real-life situation, which reinforced our understanding further.
- Good encoding is an important aspect of GAs. Good encoding helps us apply GAs to non-trivial optimization problems as well.

REFERENCES

- *The evolution of cooperation*, Robert Axelrod
- Dr. Tim Havens CS5821 class notes
- Fundamentals of Computational Intelligence: Neural Networks, Fuzzy Systems, and Evolutionary Computation – Keller, Liu, and Fogel (1st Ed)