Jagadish Bapanapally AI18 Assignment 3

[1] You are asked to design genetic algorithm to minimize $f(x) = x^2 - 7x$ over integer variable $x \in [0, 7]$. Answer the following questions.

(1.1) How would you design chromosome?[1].
Answer:
Because genetic algorithms are implemented in a way so that we search based on evolution and so to do mutations on each individual, we have to find a way to represent each individual to do mutations. $x \in [0, 7]$ using a binary string of length 3. eg: $100(x = 4), 010(x = 2)$;

(1.2) How would you design fitness function?
Answer:
Fitness function should maximize the fitness around the value that minimizes the function $f(x)$. Because for all $x \in [0, 7]$ $f(x)$ returns negative except at 0 and it is minimum approximately at around 3.5. So modulus of the function should give maximum fitness at the value that minimizes the function. So squaring, cubing and so on of the modulus will give maximum fitness at the value that minimizes the function. I choose $|x^2 - 7x|^2$ as the fitness function.

(1.3) What if x is continuous variable in $[0, 7]$, and your are restricted to use chromosome of at most 2 bits[2]? How would you apply GA to optimize $f(x)$? Clarify the key difference, if any, between your designs for continuous variable and for integer variable.
Answer:
If x is continuous variable, then we cant represent all of x domain using binary strings. So, for the initial population just choose four 4 values from x domain and denote them to binary strings of length 2. Cross over and mutate the strings based on representations and for the next generation keep 2 most fitted values and change the representation of the other 2 variables to 2 random values in x domain. And then repeat the process

[2] Implement your proposed GA algorithm in [1] from scratch[3] in Python. Your algorithm must include a single-point crossover operator and a mutation operator. Feel free to use any library or embedded function to generate any probability when needed. Let n be the population size, p_c be the cross-over probability and p_m be the mutation probability. Report your results in the following forms:

(2.1) Plot the performance graph of your implementation with $n = 6$, $p_c = 0.7$ and $p_m = 0.01$; the graph should include enough number of generations to evidence the convergence of your algorithm; properly label both axes and both curves; an example graph is in Figure 1 (left).

(2.2) Fix $n = 6$ and $p_m = 0.01$. Choose three values for p_c, i.e., 0.7, 0.2 and 0.9 and plot their average fitness in one performance graph[4]; properly label the three curves; an example graph is in Figure 1 (right).

(2.3) Fix $p_c = 0.7$ and $p_m = 0.01$. Choose two values for n, i.e., 6 and 20, and plot their average fitness in one performance graph; properly label the two curves.

[Bonus] (2.4) Implement your proposed design for the case when x is continuous variable in [0,7]. Plot the performance graph. Derive the optimal $f(x)$ in theory, and compare it with the optimal (f) achieved by your algorithm. Clarify the difference in your plotted performance graph.

---

[1] Explain the principle and give at least two example chromosomes
[2] This restriction could be due to any memory constraint of the computing device.
[3] This means you are not allowed to use any GA library or embedded function.
[4] In this graph, you do not need to plot the best fitness. Same applies to task (2.3).

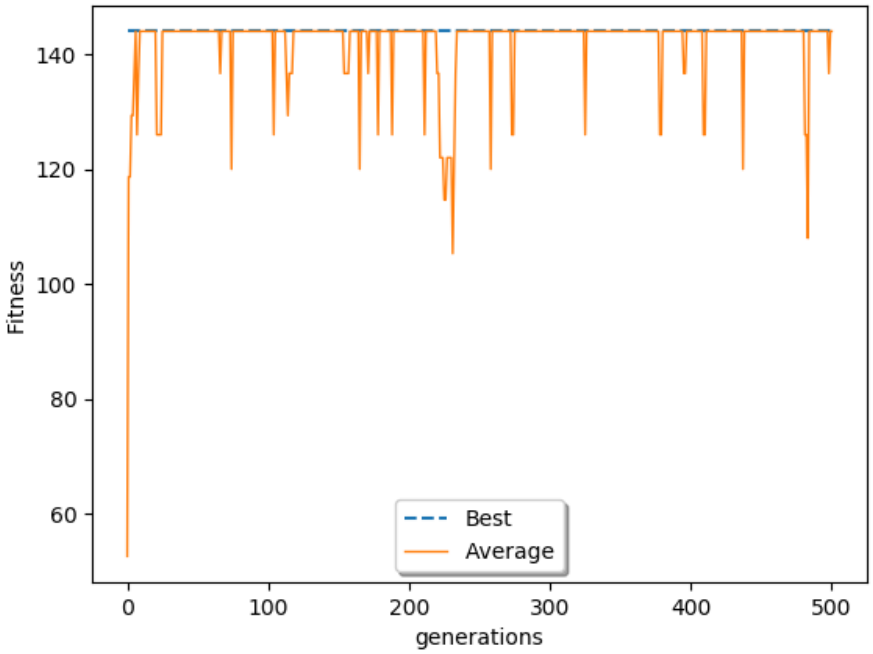

Fig. 1. best and average fitness for $pc = 0.7$

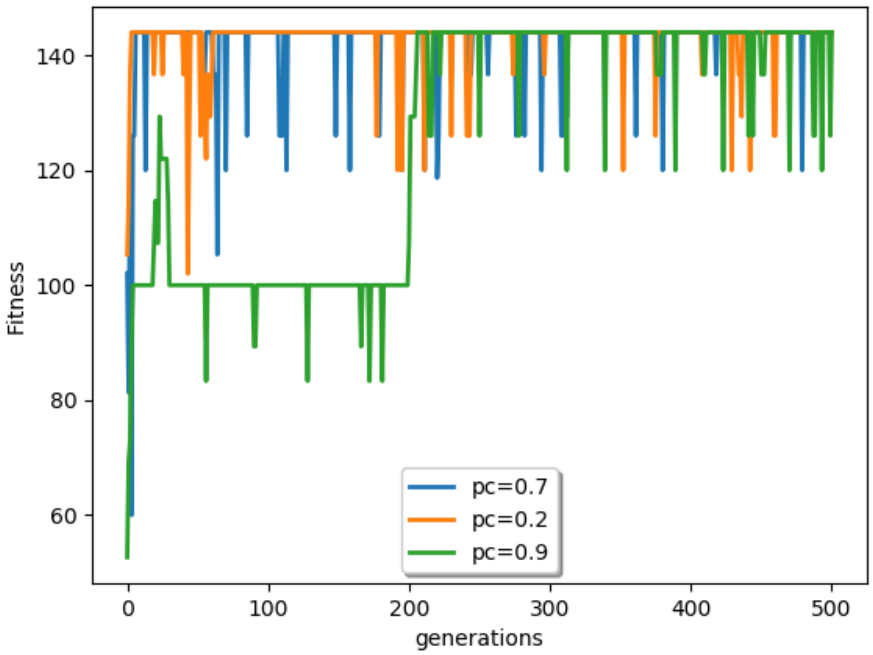

Fig. 2. 3 plots of fitness vs generations for $pc = 0.7, 0.2, 0.9$

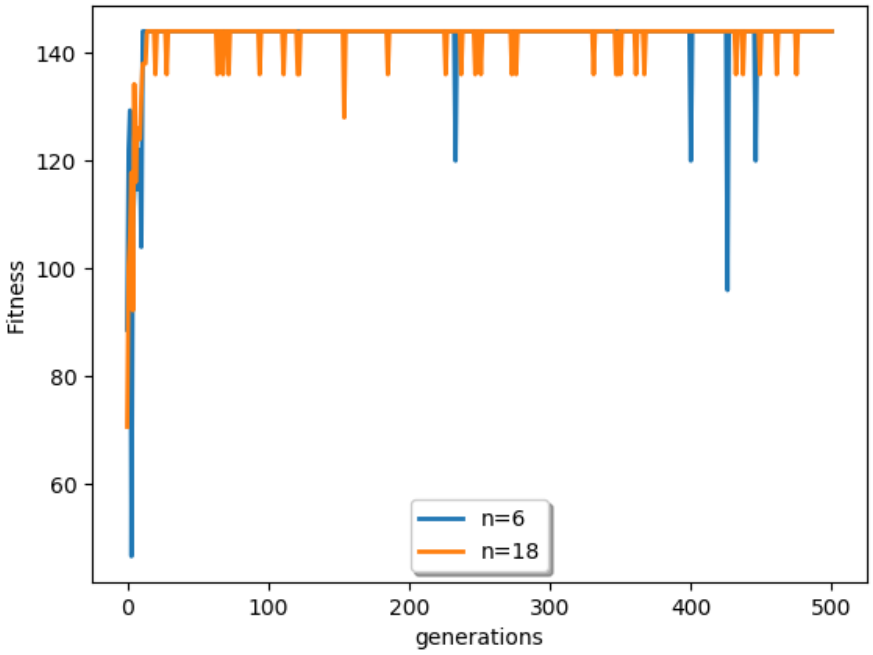

Fig. 3. 3 plots of fitness vs generations for $n = 6, 8$

[3] You are asked to design genetic algorithm to minimize function $f(x, y) = (x-5)^2 + (y-7)^2$ over integer variables $x, y \in [0, 7]$. You first encode x, y into 3-bit chromosomes separately, and let x_c, y_c respectively denote their encoded chromosomes. Then you design crossover and mutation operators. Now, you have two strategies: (a) concatenate both chromosomes into a 6-bit chromosomes $x_{xy} = [x_c, x_y]$, and apply standard crossover and mutation operators on x_{xy}; (b) separately apply standard crossover and mutation operators on x_c and y_c. You need to figure out which strategy is better.

(3.1) Implement both strategies with $n = 10$, $p_c = 0.7$ and $p_m = 0.01$. Plot their performance graphs separately, compare them, and discuss your observations and conclusions.
Answer:
Fig 4 is with doing crossover x and y together in a single string. In this method it is not sure that we will receive the optimum values for x and y in each and every run. The reason is based on the probability crossover might not occur for y.

Fig 5 is with doing crossover with x and y separately. In this method we will do crossover x and y separately so it is sure that we will get new values for y and there will be optimum values in the population.

So it is better to do crossover with x and y separately.

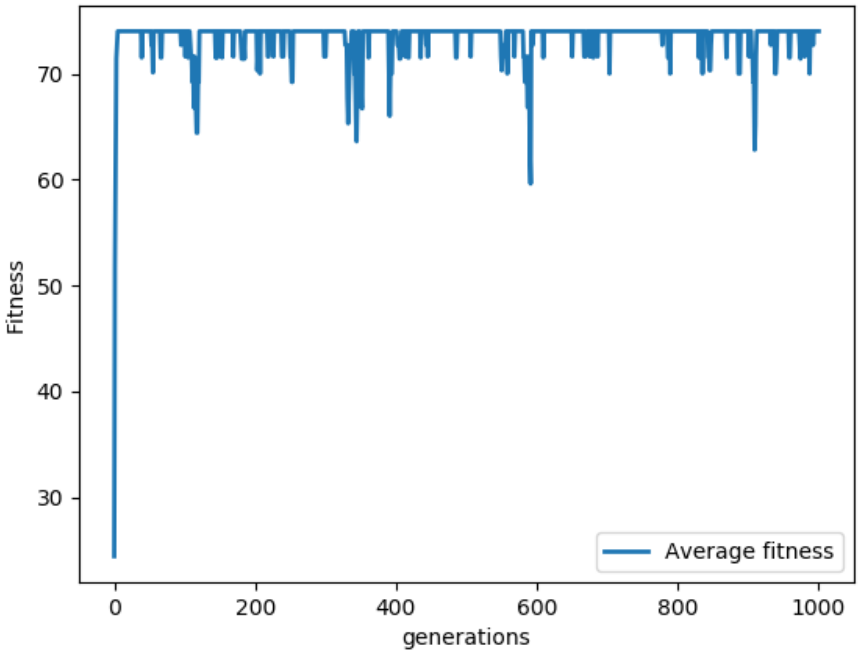

Fig. 4. 3.1(a) doing crossover for a string with x and y together

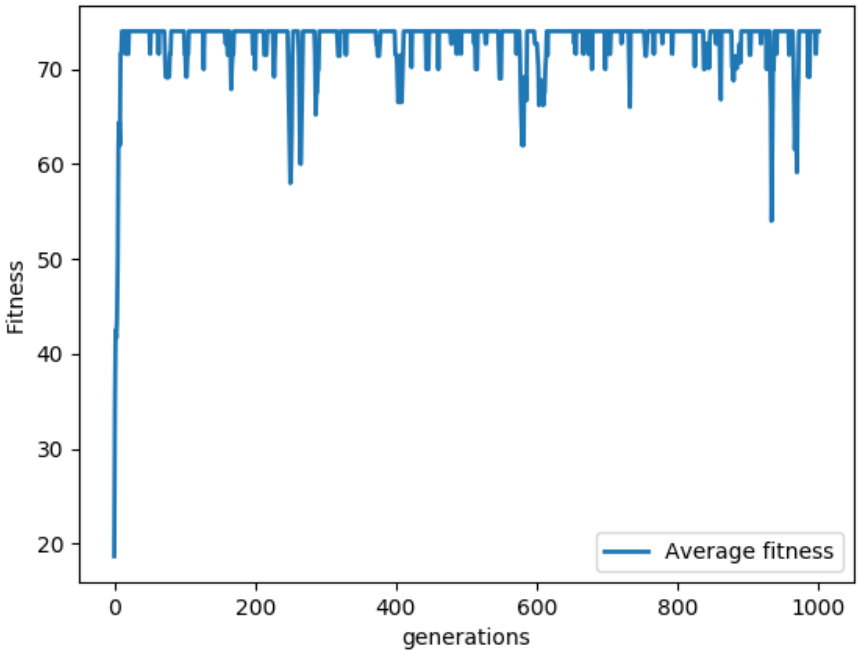

Fig. 5. 3.1(b) doing crossover for a string with x and y separately

[4] You are asked to design (1+1) evolutionary strategy with adaptive global step size based on 1/5-rule to minimize function $f(x) = x^2 - 7x$ over continuous variable $x \in [0, 7]$.

(4.1) Explain 1/5-rule and its rationale.
Answer:
If $Ps(Probabilityofsuccess) < 1/5$ then we are close to the converging point so decrease the probability mutation and if $Ps > 1/5$ then we are far from the converging point so increase the probability mutation. 1/5 is found based on experiments.
(4.2) Implement your design from scratch in Python. Separately plot performance graphs for two settings in 1/5-rule: (a) $c = 0.8$, (b) $c = 1$. Compare them and discuss your observations.

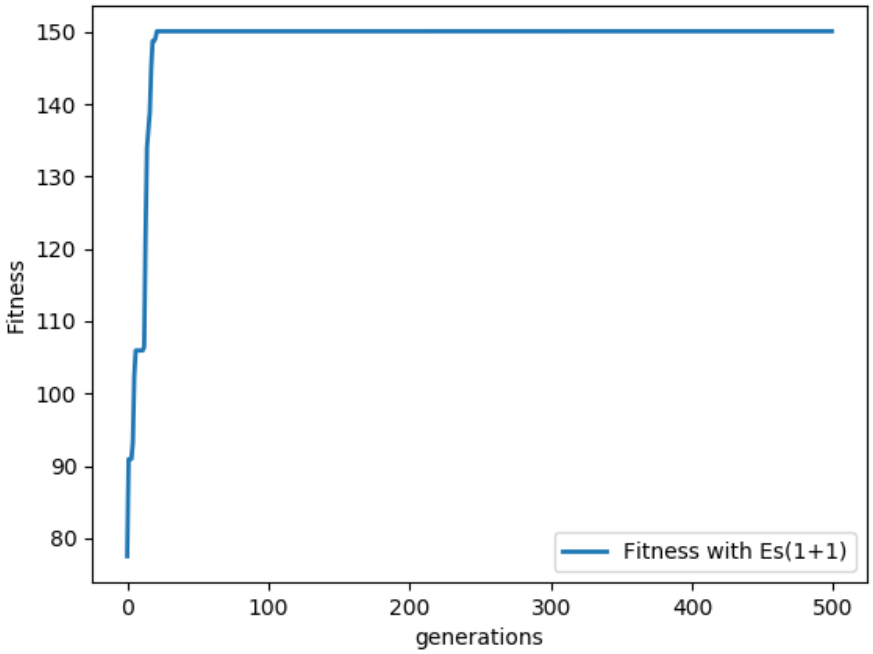

Fig. 6. 4.2(a) Fitness when c = 0.8

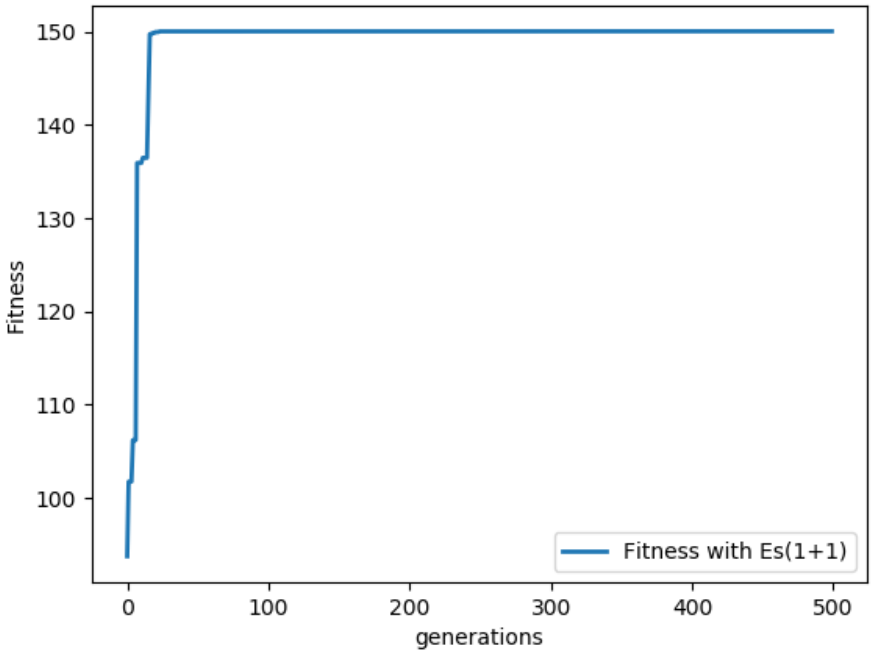

Fig. 7. 4.2(b) Fitness when c = 1

(4.3) Briefly explain the idea of recombination in evolutionary strategy. How is it different from (1+1) strategy?
Answer:
To get an offspring in evolution strategy from 2 parents we combine parameters of 2 parents. In (1+1) strategy the one offspring is obtained from 1 parent and 2nd one is from is 2nd parent.

[5] Schema Theorem is an important theoretical guarantee for genetic algorithm.

(5.1) Write down the theorem, explain every notation and the intuition of the theorem.
Answer:
A schema is a set of bit strings with ones, zeros and asterisks which represents a form of the chromosomes. The ones and zeros are the fixed positions in the chromosome and asterisks can be either ones or zeros. These string are the instances of that schema: For eg the following schema

1	*	*	0

is a set of strings with 1 in the beginning and 0 at the end. A chromosome matches a schema when the fixed positions in the schema which match the corresponding positions in the chromosome. The number of fixed bits in the schema is called the order of the schema. In the above example it is 2.
The schema theorem states that a schema with above average fitness will occur more frequently in the next generations than the schema with below average fitness. The intuition is that the schema with above average fitness will have more probability of selecting that schema for crossover and thus we will see more instances of that schema in the next generation.