GP Report - *Prime number*

1

I choose 0-n range as as my fitness function, that means I'm looking for the longest range from 0 to n can fit the prime polynomial equation $x^2 + bx + c$. I chose this as my fitness function because there is no rational algebraic function which always gives primes and Goldbach showed there is no polynomial equation with integer coefficient can show all the primes number [G. H. Hardy (1980)]. But there exits a set of primes equals the set of positive values of this polynomial. Therefore, I think that looking for a longest certain range is a good option rather than find max nb of distinct primes. I also bound the b,c [-200, 200] as initial population coefficients to improve performance.

2

Algorithm Structure:

Termination condition: Given a solution has been found or a limit of allowed generations is reached. In the end, I use 'found optimal solution' as my stop criterion.

Selection: Choosing the individuals to be used in creating the next generation. In the begin of program, I random uniformly to initialize 10 population which is saved as current population either. And choose 2 as parents based on fitness value.

IsPrime: In the loop test first thing if number is even and bigger than 2, next when range all possible divisors we add a step value, this range start at 3 and cover all odd number up to our limit.

Variation: Applying changes to selected individuals to alter their behavior. This is done by either mutation or crossover. The new offspring will have its first half of its genes taken from the first parent, second half of its genes taken from the second parent. In the $x^2 + bx + c$ we only have 2 variables, the mutation rate set to 0.5 which means how many genes in the individual will have mutation. And randomly pick up mutation position in each individual.

Pseudo Code:

```
Initial population
while not equal termination condition:
{
evaluate fitness
select best parents for mating
use crossover for next generation
adding var to offspring using mutation
assign new population
}
```

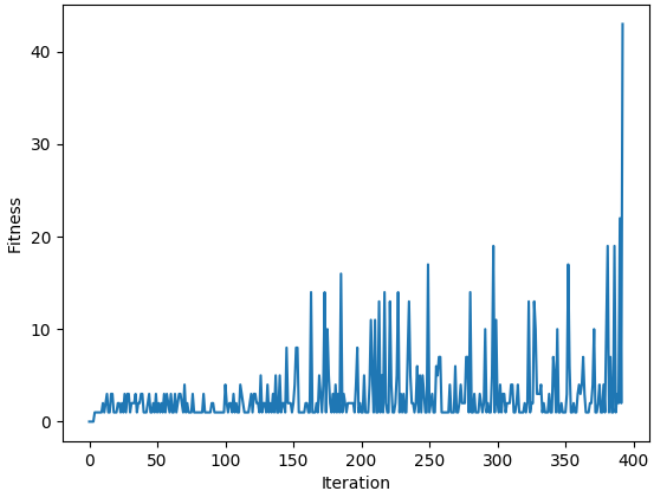
3

Bounding $x^2 + bx + c$ in [-200, 200] could converge the results more quickly. If we all bound the b, c in a positive integer. It would be hard to reach the termination condition. Because we

initial population in uniform random numbers. All positive coefficients quadratic polynomials of such form would depend on initial values and the number of prim numbers is around $log(10^n)$, therefore, have negative bound can cover more prime numbers in quadratic polynomials. When bound with Trott constants every entry can reach termination condition quickly.

Termination condition: If using a specific generation(eg. max generations) as a stop criterion may have repetition runs. The termination condition should correspond to the fitness function. In this design, I calculate the longest 0-n as my optimal solution. Therefore, using max generations is redundant. Instead, when the current population could reach an optimal solution. Then I stop the program.

4



Above the graph, we could observe the Fitness can reach 40(depend on the setting) after 200 iterations. And the overall trends are improving, that is means GA is trying to generate an optimal prime number of the polynomial equation from parents. The under table presents in the last generation the current population values, every iteration we will store optimal two value as parents through two parents crossover to generate 8 off-springs. And according to mutation rate to mutant genes in each individual. During the muting the locus is randomly and add/deduct value is either.(The list of best result see next section tablbe)

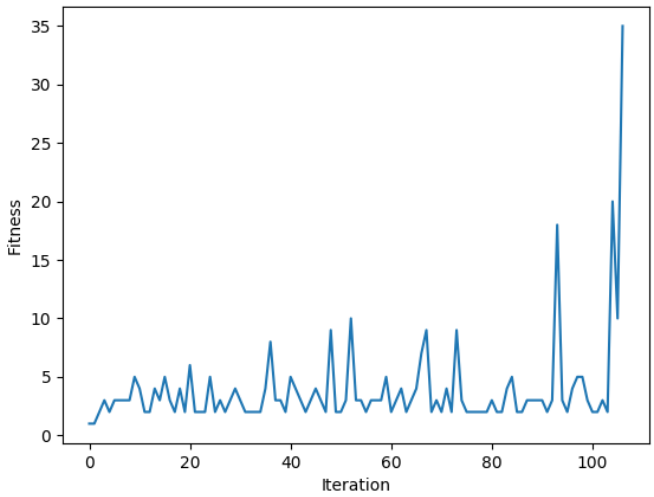
Generations	Coef B	Coef C	0-n
parent	-5	47	0-43
parent	-5	47	0-43
offspring	-5	197	0-6
offspring	148	47	0-1
offspring	46	47	0-1
offspring	181	47	0-2
offspring	-5	-136	0
offspring	-38	47	0-1
offspring	-5	140	0
offspring	67	47	0-1

5

Every run time the optimal combination is different. Because we randomly initialize the population and mutation value. The final result will be influenced by those variables. Through below we can observe that the number of generations does not have a positive correlation to the range.

Generations	Coef B	Coef C	0-n
96	13	83	0-34
422	-3	43	0-42
158	-13	83	0-47
122	-19	131	0-50

We were using distinct primes as fitness function(Below table). This method will cause a program in unlimited looping easily. Since the termination condition setting is the same as using the widest range as the fitness function. The distinct prime as fitness cause more unlimited looping, the main reason may prime distribute locates in certain range. And randomly initial values may skip this range and cause looping. Plus, the generations relays on parents, if parents are not in range and mutants rate is low it would be hard to reach termination condition(see below Fig, having good mutation which reaches termination occasionally). However, the overall generations is lower than widest. Comparing to the widest range, the distinct primes is more sensitive to termination condition and unpredictable initial values.



Generations	Coef B	Coef C	distinct primes
107	-13	83	48
47	11	71	36
60	-27	233	55
348	-31	257	33

6

My solution can fit finding the maxi-mun value from the higher degree polynomial equation. For example, we have $ax^5 + bx^4 + cx^3 + dx^2 + ex = y$ equation, and we want to find the max y from

unknown coefficients. First we can randomly generate 5 coefficients as fixed/using costumed value. And initialize 1000 populations and pick up to 10% as parents to generate off-springs. This also can apply to solve polynomial equations then the fitness function can consider the sum of all deviations squared.

Others

Env: python3 environment, install minimal python3 package.
execution: python3 prime.py

Other acknowledgement:Nikolaev and Iba (2001)

References

G. H. Hardy, E. M. W. (1980). *"an introduction to the theory of numbers"*. Oxford University Press.

Nikolaev, N., & Iba, H. (2001). Accelerated genetic programming of polynomials. *Genetic Programming and Evolvable Machines.*, *2*, 231257.