Example of Selection

Evolutionary Algorithms is to maximize the function $f(x) = x^2$ with x in the integer interval $[0, 31]$, i.e., $x = 0, 1, \ldots 30, 31$.

1. The first step is encoding of chromosomes; use binary representation for integers; 5-bits are used to represent integers up to **31**.
2. Assume that the population size is **4**.
3. Generate initial population at random. They are chromosomes or genotypes; e.g., **01101, 11000, 01000, 10011**.
4. Calculate fitness value for each individual.

   (a) Decode the individual into an integer (called phenotypes),

   01101 → 13; 11000 → 24; 01000 → 8; 10011 → 19;

   (b) Evaluate the fitness according to $f(x) = x^2$,

   13 → 169; 24 → 576; 8 → 64; 19 → 361.
5. Select parents (two individuals) for crossover based on their fitness in p_i. Out of many methods for selecting the best chromosomes, if **roulette-wheel** selection is used, then the probability of the i^{th} string in the population is $p_i = F_i \,/\, (\sum_{j=1}^{n} F_j)$, where

   F_i is fitness for the string i in the population, expressed as $f(x)$

   p_i is probability of the string i being selected,

   n is no of individuals in the population, is population size, $n=4$

   $n * p_i$ is expected count

String No	Initial Population	X value	Fitness Fi $f(x) = x^2$	p i	Expected count N * Prob *i*
1	0 1 1 0 1	13	169	0.14	0.58
2	1 1 0 0 0	24	576	0.49	1.97
3	0 1 0 0 0	8	64	0.06	0.22
4	1 0 0 1 1	19	361	0.31	1.23
Sum			1170	1.00	4.00
Average			293	0.25	1.00
Max			576	0.49	1.97

The string no 2 has maximum chance of selection.

- **Example 1 :**

  Maximize the function $f(x) = x^2$ over the range of integers from **0 . . . 31**.

  Note : This function could be solved by a variety of traditional methods such as a hill-climbing algorithm which uses the derivative.

  One way is to :

  - Start from any integer **x** in the domain of **f**
  - Evaluate at this point **x** the derivative **f′**
  - Observing that the derivative is **+ve**, pick a new **x** which is at a small distance in the **+ve** direction from current **x**
  - Repeat until **x = 31**

  See, how a genetic algorithm would approach this problem ?

1. Devise a means to represent a solution to the problem :

   Assume, we represent **x** with five-digit unsigned binary integers.

2. Devise a heuristic for evaluating the fitness of any particular solution :

   The function **f(x)** is simple, so it is easy to use the **f(x)** value itself to rate the fitness of a solution; else we might have considered a more simpler heuristic that would more or less serve the same purpose.

3. Coding - Binary and the String length :

   GAs often process binary representations of solutions. This works well, because crossover and mutation can be clearly defined for binary solutions. A Binary string of length **5** can represents 32 numbers (0 to 31).

4. Randomly generate a set of solutions :

   Here, considered a population of four solutions. However, larger populations are used in real applications to explore a larger part of the search. Assume, four randomly generated solutions as : **01101, 11000, 01000, 10011.** These are chromosomes or genotypes.

5. Evaluate the fitness of each member of the population :

   The calculated fitness values for each individual are -

   (a) Decode the individual into an integer (called phenotypes),

   01101 → 13; 11000 → 24; 01000 → 8; 10011 → 19;

   (b) Evaluate the fitness according to **f(x)** = x^2 ,

   13 → 169; 24 → 576; 8 → 64; 19 → 361.

   (c) Expected count = **N * Prob i** , where **N** is the number of individuals in the population called population size, here **N = 4.**

   Thus the evaluation of the initial population summarized in table below .

String No i	Initial Population (chromosome)	X value (Pheno types)	Fitness $f(x) = x^2$	Prob i (fraction of total)	Expected count N * Prob i
1	0 1 1 0 1	13	169	0.14	0.58
2	1 1 0 0 0	24	576	0.49	1.97
3	0 1 0 0 0	8	64	0.06	0.22
4	1 0 0 1 1	19	361	0.31	1.23
Total (sum)			1170	1.00	4.00
Average			293	0.25	1.00
Max			576	0.49	1.97

Thus, the string no 2 has maximum chance of selection.

6. Produce a new generation of solutions by picking from the existing pool of solutions with a preference for solutions which are better suited than others:

We divide the range into four bins, sized according to the relative fitness of the solutions which they represent.

Strings	*Prob i*	*Associated Bin*
0 1 1 0 1	0.14	0.0 . . . 0.14
1 1 0 0 0	0.49	0.14 . . . 0.63
0 1 0 0 0	0.06	0.63 . . . 0.69
1 0 0 1 1	0.31	0.69 . . . 1.00

By generating **4** uniform **(0, 1)** random values and seeing which bin they fall into we pick the four strings that will form the basis for the next generation.

Random No	*Falls into bin*	*Chosen string*
0.08	0.0 . . . 0.14	0 1 1 0 1
0.24	0.14 . . . 0.63	1 1 0 0 0
0.52	0.14 . . . 0.63	1 1 0 0 0
0.87	0.69 . . . 1.00	1 0 0 1 1

7. Randomly pair the members of the new generation

Random number generator decides for us to mate the first two strings together and the second two strings together.

8. Within each pair swap parts of the members solutions to create offspring which are a mixture of the parents :

For the first pair of strings: **0 1 1 0 1 , 1 1 0 0 0**

– We randomly select the crossover point to be after the fourth digit. Crossing these two strings at that point yields:

0 1 1 0 1 ⇒ 0 1 1 0 |1 ⇒ 0 1 1 0 0

1 1 0 0 0 ⇒ 1 1 0 0 |0 ⇒ 1 1 0 0 1

For the second pair of strings: **1 1 0 0 0 , 1 0 0 1 1**

– We randomly select the crossover point to be after the second digit. Crossing these two strings at that point yields:

1 1 0 0 0 ⇒ 1 1 |0 0 0 ⇒ 1 1 0 1 1

1 0 0 1 1 ⇒ 1 0 |0 1 1 ⇒ 1 0 0 0 0

9. Randomly mutate a very small fraction of genes in the population :

With a typical mutation probability of per bit it happens that none of the bits in our population are mutated.

10. Go back and re-evaluate fitness of the population (new generation) :

This would be the first step in generating a new generation of solutions. However it is also useful in showing the way that a single iteration of the genetic algorithm has improved this sample.

String No	*Initial Population (chromosome)*	*X value (Pheno types)*	*Fitness $f(x) = x^2$*	*Prob i (fraction of total)*	*Expected count*
1	0 1 1 0 0	12	144	0.082	0.328
2	1 1 0 0 1	25	625	0.356	1.424
3	1 1 0 1 1	27	729	0.415	1.660
4	1 0 0 0 0	16	256	0.145	0.580
Total (sum)			1754	1.000	4.000
Average			439	0.250	1.000
Max			729	0.415	1.660

Observe that :

1. Initial populations : At start step 5 were

   0 1 1 0 1 , 1 1 0 0 0 , 0 1 0 0 0 , 1 0 0 1 1

   After one cycle, new populations, at step 10 to act as initial population

   0 1 1 0 0 , 1 1 0 0 1 , 1 1 0 1 1 , 1 0 0 0 0

2. The total fitness has gone from **1170** to **1754** in a single generation.
3. The algorithm has already come up with the string 11011 (i.e **x = 27**) as a possible solution.

+ operator mutation from our class…