EVOLUTIONARY COMPUTATION

GENETIC ALGORITHMS

Evolutionary Computation

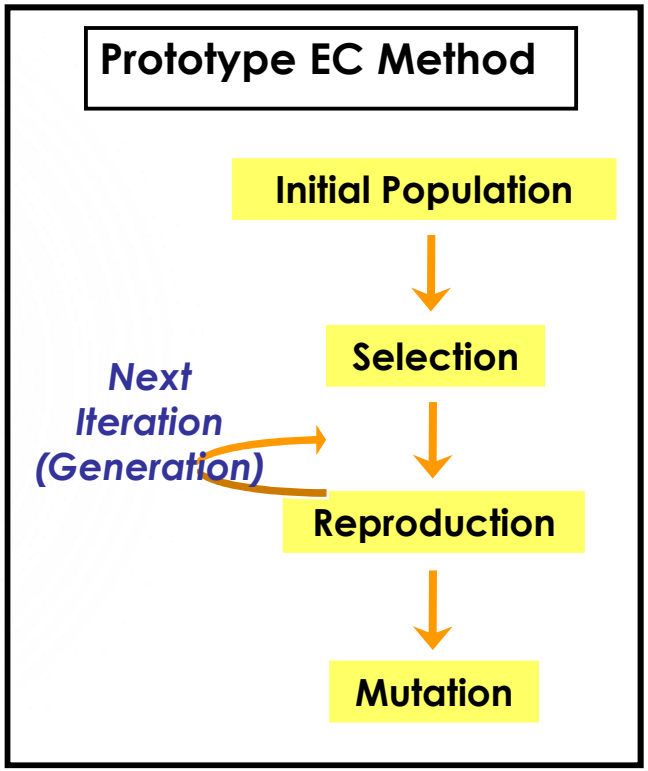
Optimization problems could have ONE or MORE solution(s)

- Evolutionary computation (EC) methods are based on ***population*** of solutions
- Every individual in the population represents one solution
- Solutions from EC could be near or far from desired values
  - Each iteration involves updating all elements of the population
  - Each member in the population corresponds a fitness value
- Three **famous** types of EC
  - Genetic Algorithms
  - Particle Swarm
  - Ant Colony
- Genetic algorithm (GA) is the scope of this lecture

Genetic Algorithms

Genetic Algorithms (GAs) are **meta-heuristic** search algorithms based on the evolutionary ideas of natural genetics.

GA is an **Evolutionary computing** method inspired about **survival of the fittest**



General idea of GA

- Starts from an initial population of randomly generated
- Each generation:
  - The fitness of every individual in the population is evaluated
  - A group of individuals is selected from the current population (based on their fitness) to continue …
  - Modified individuals are generated from the rest
- A new population is thus created

Fitness Function and Coding of θ

- Need to define "fitness function" to be maximized
- The fitness is inversely proportional to the errors between desired values and the θ values of all the population
- Every iteration, extreme values should be cancelled
- Coding of θ:
  - Bit representation applies to each element of θ for each of the members of the population (e.g., $\theta \rightarrow$ [0 1 1 0…1 0])
  - Real-number "coding" (i.e., no coding of θ) becoming popular due to effectiveness in applications

GA Steps

Algorithm for GA

Initialization

- An initial population is generated randomly
- The population size depends on the nature of the problem
- Population contains several hundreds or thousands of possible solutions
- The population covers the entire range of possible solutions

Algorithm for GA

Selection

- During each successive generation, a group of individuals is selected to breed a new generation
- Selection is based on a *fitness function*
- Fitness functions are Stochastic and should:-
  - Permit most fitting solutions to be selected
  - Prevent convergence on poor solutions
  - keep the diversity of the population large
- **Example:** Roulette wheel selection fitness function
  - Individuals are given a probability of being selected directly proportionate to their fitness
  - Individuals for a new generation are chosen based on these probabilities

Algorithm for GA

Reproduction

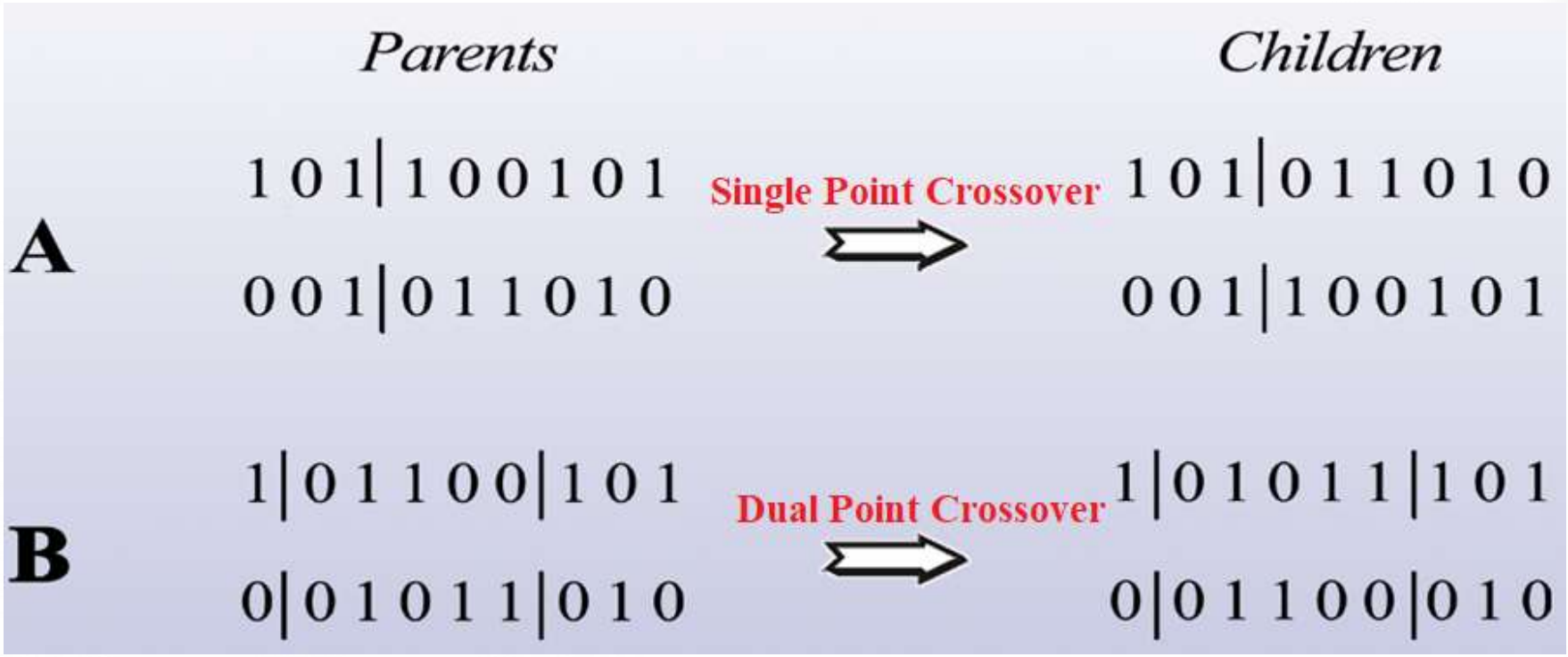
- The next step is to generate a 2nd generation (population) through genetic operators:
  - Crossover
  - Mutation
- Reproduction is achieved through a pair of "parents" to produce a "child"
- This "child" typically shares many of the characteristics of its "parents"
- The process is then repeated

Algorithm for GA

Reproduction: Crossover

- In single point crossover, you choose a locus at which you swap the remaining bits from on parent to the other
- The children takes one section of the chromosome from each parent
- The locus or crossover point is selected randomly
- This particular method is called single point crossover
- For two locus, there could be dual point crossover
- The probability of crossover occurring is usually 60% to 70%, however, sometimes no crossover occurs (0%)

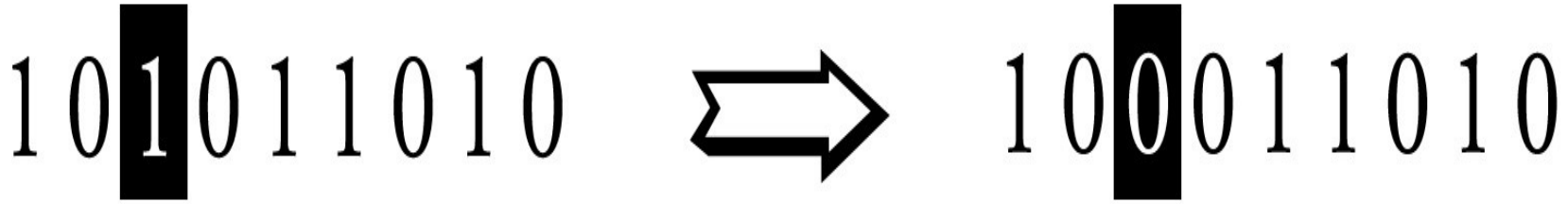
Crossover


Parents
Children
A
1 0 1 | 1 0 0 1 0 1
0 0 1 | 0 1 1 0 1 0
Single Point Crossover
1 0 1 | 0 1 1 0 1 0
0 0 1 | 1 0 0 1 0 1
B
1 | 0 1 1 0 0 | 1 0 1
0 | 0 1 0 1 1 | 0 1 0
Dual Point Crossover
1 | 0 1 0 1 1 | 1 0 1
0 | 0 1 1 0 0 | 0 1 0

Algorithm for GA

Reproduction: Mutation

- In order to ensure that the individuals are not all exactly the same, you allow for a small chance of mutation
- You loop through all the bits of all the individuals, you can either change it by a small amount or replace it with a new value
- Usually this is applied once / individual
- The probability of mutation is usually between 0% and 20%



Algorithm for GA

Termination

- Common terminating conditions are:
  - A solution is found
  - Fixed number of generations is reached
  - Allocated budget (computation time/money) is reached
  - Successive iterations no longer produce better results
  - Any Combinations of the above

Pseudo-code

BEGIN

INITIALIZE population with random candidate solutions

EVALUATE fitness for each candidate;

Exclude weakest and magnify fittest;

REPEAT UNIT (termination condition) is satisfied DO

1. SELECT parents;
2. CROSSOVER pairs of parents;
3. MUTATE the resulting offspring;
4. SELECT individuals or the next generation;

END.

Encoding

Chromosomes could be:

- Bit strings (0101 ... 1100)
- Real numbers (43.2 -33.1 ... 0.0 89.2)
- Permutations of element (E11 E3 E7 ... E1 E15)
- Lists of rules (R1 R2 R3 ... R22 R23)

Very important to think about a good manner to code your individuals … coding intelligence affects the whole process

Binary Encoding

One variable function, say 0 to 15 numbers, numeric values, represented by 4 bit binary string.

Numeric value	4-bit string	Numeric value	4-bit string	Numeric value	4-bit string
0	0 0 0 0	6	0 1 1 0	12	1 1 0 0
1	0 0 0 1	7	0 1 1 1	13	1 1 0 1
2	0 0 1 0	8	1 0 0 0	14	1 1 1 0
3	0 0 1 1	9	1 0 0 1	15	1 1 1 1
4	0 1 0 0	10	1 0 1 0		
5	0 1 0 1	11	1 0 1 1		

GA: Advantages

1. A robust (solid) search technique
2. Simple to develop
3. Easy to incorporate with other methods
4. Solutions are understandable
5. Provide many alternative solutions
6. Performance (accuracy) is proportional to computational cost

GA: Disadvantages

1. No guarantee for an optimal solution
2. Weak theoretical basis
3. Parameter tuning could have unlimited scenarios
4. Computationally expensive when high accuracy is needed

Example 1 :

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

The problem desired solution (**integer number ranging between 0 and 31**)

A binary string of 5 bits will be the solution ranging from 00000 to 11111

1. Select the fitness function: The most easy way is to use the main function f(x)
2. Select the coding method: We will use binary coding for all algorithms
3. Start the GA process

   Initialization

   We can use a population of 4

01101	11000	01000	10011 (randomly selected)
13	24	8	19

Evaluate the fitness of each member of the population :

The calculated fitness values for each individual are -

(a) Decode the individual into an integer

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) Calculate the probability of each individual

$\frac{169}{1170} = 0.14$ $\frac{576}{1170} = 0.49$ $\frac{64}{1170} = 0.06$ $\frac{361}{1170} = 0.31$

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

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

Selection

- String (3) has the least probability, this should be omitted from the population
- String (2) has the highest probability, this will be reproduced
- The new population will be:

Chosen string
0 1 1 0 1
1 1 0 0 0
1 1 0 0 0
1 0 0 1 1

Reproduction

Randomly pair the members of the new generation

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

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

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

The new generation is: 01100 11001 11011 10000

Randomly mutate a very small fraction of genes in the population : none of the bits in our population are mutated.

to facilitate understanding the process

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)
1	0 1 1 0 0	12	144	0.082
2	1 1 0 0 1	25	625	0.356
3	1 1 0 1 1	27	729	0.415
4	1 0 0 0 0	16	256	0.145
Total (sum)			1754	1.000
Average			439	0.250
Max			729	0.415

Observations

Initial Population (Chromosome)	Integer Value	2nd Iteration (Chromosome)	Integer Value
01101	13	01100	12
11000	24	11001	25
01000	8	11011	27
10011	19	10000	16

After only one iteration, the population is updated towards better values

Remember that the exact solution is 31 = 11111

The total fitness value has changed from 1170 to 1754

The algorithm has come up with X=27 (nearby the exact solution)

Repeating for another iteration, this will goes towards better population and better solution near exact

Practical Example

Knapsack Problem

- Given **N** items where each item has some weight and volume associated
- Given a bag with capacity **W**, [i.e., the bag can hold at most **W** weight in it]
- The task is to put the maximum number of items into the bag such that the sum of volumes is the maximum possible

Note: It is not possible to put a part of an item into the bag

https://youtu.be/uQj5UNhCPuo

Questions

- From your evolutionary computation study, explain what is meant by survival of fittest. You can use graph – chart – pseudocde
- Write down the pseudo-code of the GA.
- Could you explain the difference between "crossover" and "mutation" in GA?
- What are two main advantages and disadvantages of GA?

Thank You