AI Searching Techniques

Computational Intelligence

Dr.Sherif Saad

Last-Time

- Introduction to Computational Intelligence
- Computational Intelligence Paradigms

Learning Objectives

- Introduce the students to the theory of Evolutionary Computation
- Understand the main concepts and terminologies of genetic algorithm
- Learn how to apply genetic algorithm to solve problems
- Learn how to implement genetic algorithm

Outlines

1. Introduction to Evolutionary Computation and Genetic Algorithms
2. Genetic Algorithm Population
3. Genetic Algorithm Encoding
4. Genetic Operators
5. Problem Solving with Genetic Algorithm
6. Genetic Algorithm Implementation in Python

Genetic Algorithm: Fitness Function

Takes a candidate solution as an input and produces as output that we can use to judge how good is the solution.

We calculate the fitness function for each individual in the population and in each generation.

A good fitness function should be fast to compute (minimum time and space complexity)

A good fitness function sensitive enough to distinguish between different candidate solutions

Genetic Algorithm: Genetic Operators

There are three main genetic operators, namely:

- Selection: How do we select parents from the population for reproduction
- Mutation: How do tweak the chromosome to introduce new genetic materials
- Crossover: How do we generate new offspring from selected parents

Genetic Operators: Selection

The overall rule for parents selection is survival of the fittest.

The best individuals should survive and create new offsprings

Proportional Selection: The best individuals should survive and create new offsprings. In the original version of GAs, individuals are chosen to mate with probability proportional to their fitness.

There are may methods for selecting the best chromosomes and maintain the key properties of natural selection and evolution theory. For example, random selection, tournament selection, rank selection, and Boltzmann selection

Genetic Operators: Selection

Proportionate Selection

- In general there are two main methods to implement Proportionate Selection, these methods are roulette wheel selection and stochastic universal sampling.
- In proportionate selection, every individual in the population can be selected for reproduction with a probability which is proportional to its fitness.The best individual has a better chance to be selected and propagate their good characteristics to the next generation.
- The approach in general *does not totally eliminate the week individuals from mating but reduce their chances*.

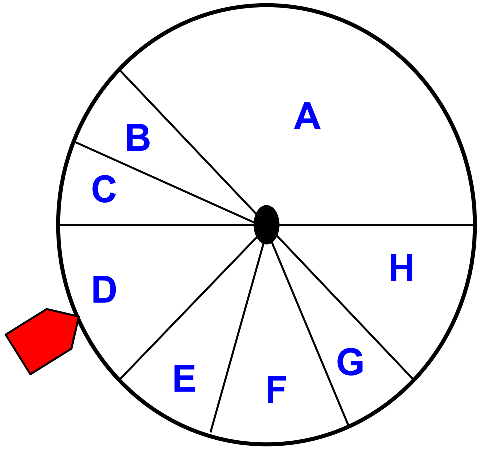
Genetic Operators: Selection

Roulette Wheel Selection

- It is a fitness proportionate selection method. Where we have a circular wheel the wheel is divided into n number of pies, n is the number of individuals in the population.
- The size of each pie in the circular wheel is proportional to the fitness of the individual represented by this pie. The bigger the fitness the bigger the pie size.
- A fixed point on the wheel circumference is selected, we rotate the wheel and the pie that comes in front of the fixed point is chosen for reproduction

Roulette Wheel Selection: Example

Chromosome	Fitness
A	40
B	5
C	7
D	18
E	10
F	15
G	10
H	20



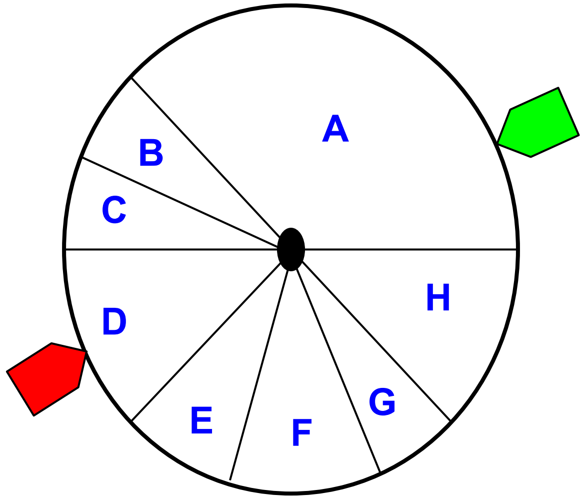
Roulette Wheel Selection: Algorithm

Can you think of an algorithm to implement the Roulette Wheel Selection.

Genetic Operators: Selection

Stochastic Universal Sampling

- It is a fitness proportionate selection method.
- It is similar to the roulette wheel selection. However, the wheel has multiple fixed selection points.
- What is the advantages of this approach or over roulette wheel selection?



Genetic Operators: Selection

Tournament Selection

- Different selection method that is fitness proportional selection.
- We select randomly m number of individuals from the population
- Then out of the m selected candidates we select the best (fittest) one of them.
- We repeat the process to select the next parent

Chromosome	Fitness
A	40
B	5
C	7
D	18
E	10
F	15
G	10
H	20

$H \left\{ \begin{array}{l} B \\ \vdots \\ H \end{array} \right.$

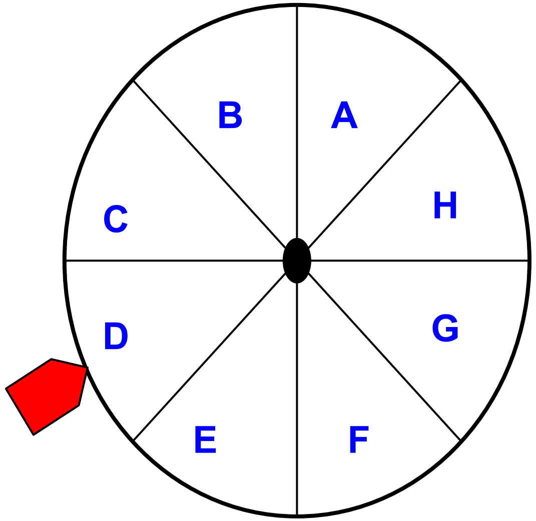
Genetic Operators: Selection

Ranked Selection

- Commonly used when the individual in the population has very similar fitness values. When do you think this happen?
- In this case the proportional selection is not useful since every individual in the population has almost the same probability to be selected.
- We rank (sort) the individuals in the population based on their fitness and the one with the highest rank is selected for reproduction.

Ranked Selection: Example

Rank	Chromosome	Fitness
1	A	25.8
2	B	25.5
3	C	25.0
4	D	24.5
5	E	24.1
6	F	23.4
7	G	23.2
8	H	23.0



Genetic Operators: Selection

Random Selection

- In this method we randomly select individual from the current population for reproduction regardless of their fitness or rank
- Each individual has the same probability to be selected.
- What are the pro and cons of this approach?

Genetic Operators: Crossover

The selected individuals (parents) for reproduction go through a crossover operation to generate one or more offspring.

The new offspring share genetic material of its parent(s).

There are three main types of crossover:

- Asexual: where an offspring is generated from one parent.
- Sexual: where two parents are used to produce one or two offspring.
- Multi-Recombination: where more than two parents are used to produce one or more offspring.

Genetic Operators: Crossover

The chromosome encoding play a major role in the implementation of the crossover.

The GA designer might implement|desing a problem-specific crossover operator

We will cover the following crossover operators:

- Binary Crossover Operators
- Value Crossover Operators
- Permutation Crossover Operators

Genetic Operators: Crossover

Binary Encoding Crossover

- Traditionally there are two methods to implement crossover
  - One Point Crossover: we select a random crossover point and the tails of its two parents are swapped to get the new off-springs. The result is two new offsprings
  - Multi Point Crossover: multiple random crossover points are selected and the segments between them are swapped to get the new off-springs.

Genetic Operators: Crossover

One Point Crossover

Chromosome (Parent A)	1 0 1 1 0 1 1 0 ¦ 0 0 0 1 1 0 0 1 1
Chromosome (Parent B)	1 1 0 0 1 1 1 0 ¦ 1 1 1 1 1 0 0 0 0
Offspring AB	1 0 1 1 0 1 1 0 1 1 1 1 1 0 0 0 0
Offspring BA	1 1 0 0 1 1 1 0 0 0 0 1 1 0 0 1 1

Genetic Operators: Crossover

Multi-Point Crossover

Chromosome (Parent A)	1 0 1 1 0	1 1 0 0 0 0 1	1 0 0 1 1
Chromosome (Parent B)	1 1 0 0 1	1 1 0 1 1 1 1	1 0 0 0 0
Offspring AB	1 0 1 1 0	1 1 0 1 1 1 1	1 0 0 1 1
Offspring BA	1 1 0 0 1	1 1 0 0 0 0 1	1 0 0 0 0

Genetic Operators: Crossover

Uniform Crossover

- This is very common in value encoding chromosome.
- We treat each gene independently and we decide which gene from which parent we are going to include in the offspring.
- It is also possible to decide that the genes from one parent will dominate the offsprings.
- The selection of the genes that will be copied to the offspring could be random or problem specific.

Genetic Operators: Crossover

Uniform Crossover

Chromosome (Parent A)	2 3 6 8 1 11 7 9 4 3 12 10 0 14
Chromosome (Parent B)	6 5 11 17 -5 12 9 21 3 -7 11 19 9 1
Offspring AB	2 5 6 8 17 -5 7 9 4 21 12 -7 0 14
Offspring BA	6 3 11 17 1 12 4 21 12 -7 0 19

Genetic Operators: Crossover

Permutation Crossover Operators

- It is very important to maintain the order of the solution when dealing with a permutation based chromosome.
- Select two random crossover points in the parent chromosomes, copy one segment from the first parent to the first offspring.
- Starting from the second crossover point in the second parent, copy the remaining genes without duplication to the first offspring, wrapping around the genes in the first offspring
- Reverse the parents order and repeat the process to generate the second offspring

Genetic Operators: Crossover

Permutation Crossover Operators

Chromosome (Parent A)	4	7	1	0	9	3	5	6	2	8
Chromosome (Parent B)	1	5	7	0	9	4	3	2	8	6
Offspring AB	5	7	4	0	9	3	2	8	6	1

Genetic Operators: Mutation

The objective of the mutation operator is to introduce new genetic material into the an existing individual.

We select some gene randomly and change their values.

Mutation is applied to some genes of the offspring with certain probability

The mutation rate is a small value between [0,1].

What do you think the importance of mutation?

What is more important mutation or crossover?

Genetic Operators: Mutation

The design and the implementation of the mutation is usually problem specific and it is up to the GA designer.

Do you think the chromosome encoding affect the design of the mutation operator?

There are some common mutation operators implementation, such as:

- Bit Flip Mutation
- Swap Mutation
- Scramble Mutation
- Inversion Mutation

Genetic Operators: Mutation

Bit Flip Mutation

Common in binary encoding (but not limited) works with discrete features

We select one or more random gene and flip their value

Offspring AB 1 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1 1

Mutation

Offspring AB 1 0 0 1 0 1 0 0 0 0 1 1 1 1 0 1 1

Genetic Operators: Mutation

Swap Mutation

We randomly select two or more genes and interchange their values.

This is commonly applied with permutation encoding

Offspring AB 2 4 5 7 8 1 9 3 6 0

Mutation

Offspring AB 2 9 5 7 8 1 4 3 6 0

Genetic Operators: Mutation

Scramble Mutation

We select a partition of the chromosome and we reorder or scramble the genes in this partition

This is commonly applied with permutation encoding

Offspring AB 2 4 5 7 8 1 9 3 6 0

Mutation

Offspring AB 2 7 4 8 5 1 9 3 6 0

Genetic Operators: Mutation

Inversion Mutation

We select a partition of the chromosome and we invert the entire partition

This is commonly applied with permutation encoding

Offspring AB   2 4 5 7 8 1 9 3 6 0

Mutation

Offspring AB   2 8 7 5 4 1 9 3 6 0

Genetic Operators: Mutation

Random Mutation

We select one or more random gene and assigned it a new value from a set of allowed values

This is commonly applied with value encoding

Offspring AB Up Up Left Up Down Right Left

Mutation ↓

Offspring AB Up Right Left Up Up Right Left

Genetic Algorithms: Termination Conditions

When the GA should terminate or stop?

- The GA performance and progress is usually very fast at the beginning. It will come with better solutions in few iterations.
- Then it will start to slow down (there is no great improvement in the solution fitness anymore from one iteration to the next) **why?**
- Our objective is to **terminate** the GA with a ***solution close to the optimal***

Genetic Algorithms: Termination Conditions

In general we can decide to terminate the GA when:

- When there has been no improvement in the population fitness for n number of iterations.
- After a predefined number of generations
- When we reach specific fitness value, or get a solution with certain characteristics.

The termination conditions are also problem specific and it is up to the GA designer to decide the termination condition that fit it the problem he is trying to solve.

Questions