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

Evolutionary Computation

Evolutionary Computation (EC) refers to computer-based problem-solving systems that use computational models of evolutionary processes, such as natural selection, survival of the fittest and reproduction

The Darwinian Theory of Evolution:

- In a world with limited resources and stable populations, each individual competes with others for survival. individuals with the “best” characteristics (traits) are more likely to survive and to reproduce, and those characteristics will be passed on to their offspring.
- During the production of a child organism, random events cause random changes to the child organism’s characteristics.

Generic Evolutionary Algorithm

- An evolutionary algorithm (EA) is a stochastic search for an optimal solution to a given problem.
- The main components of an EA algorithm:
  - Chromosome Presentation: an encoding of solutions to the problem as a chromosome
  - Fitness Function: a function to evaluate the fitness, or survival strength of individuals.
  - Initialization Operation: the process of initializing the population
  - Selection Operator: decide which individual will go through reproduction.
  - Reproduction Operator: create offsprings in every generation

Generic Evolutionary Algorithm

```
Let t = 0 be the generation counter
Create and initialize an initial population P to consist of n individuals

while stopping condition(s) not True:
    Evaluate the fitness of each individual i in P
    Perform reproduction to create offspring
    Select the new population P(t+1)
    Advance to the new generation
    t = t + 1
end
```

Genetic Algorithms: Overview

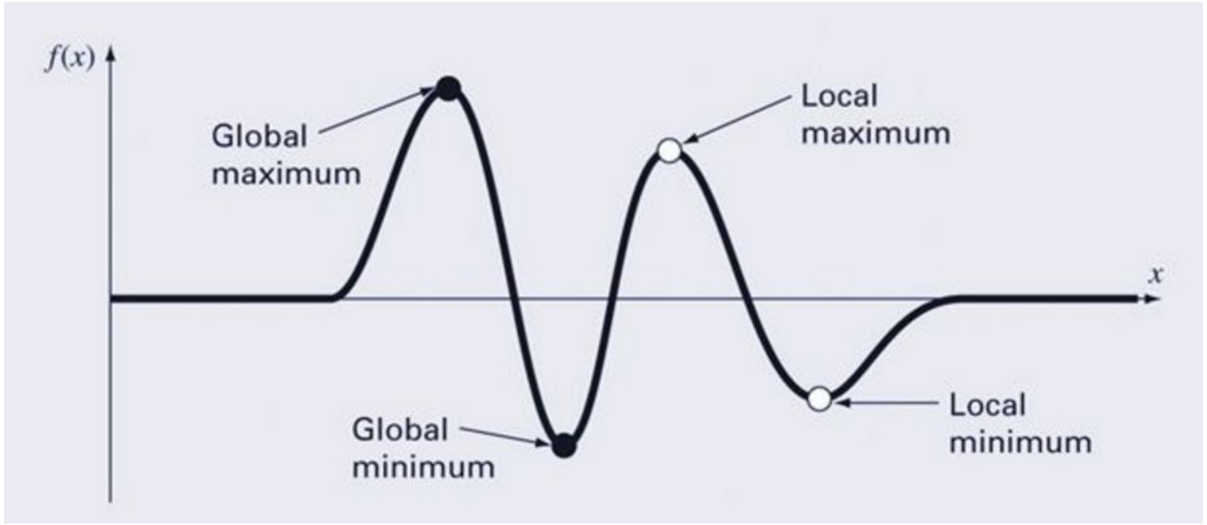
- Genetic algorithms are evolutionary computing techniques developed by John Holland at Michigan University
- They apply the principle of "survival of the fittest"
- They model the search problem as an evolution process.
- It is a search-based optimization technique inspired by the natural selection theory.
- We use genetic algorithm to find the optimal or near the optimal solution.

Genetic Algorithm Components

- Population: is a set of possible solutions (bad and good solutions). We usually use the term individuals to refer to the solutions
- Encoding Method: solutions or individuals are represented in a structure that fit the problem domain.
- Fitness Function: measure how fit (good) a solution in the population for the problem.
- Genetic Operators: Selection, Mutation and Crossover operation that define the offspring
- Termination Condition: a set of conditions to determine when the GA will stop

Genetic Algorithm: Motivation

Ability to find good enough solution for complex NP-Hard problems in reasonable time where traditional techniques usually fail.



Basic Genetic Algorithm Vocabularies

Population: is a set of possible solutions of a given problem

Chromosomes: one candidate solution in the population

Gene: one element in the chromosome

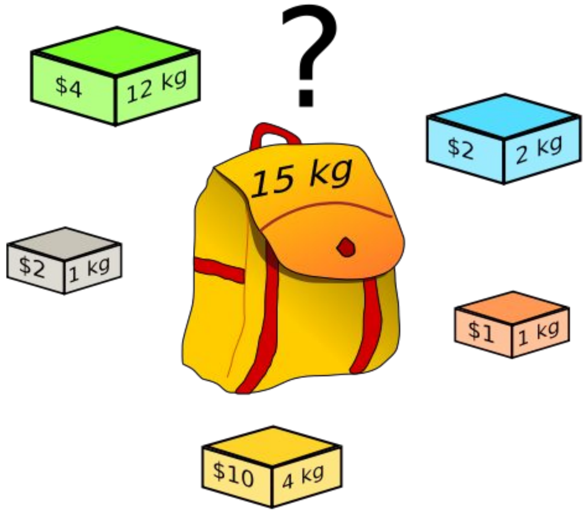
Allele: a value of a gene in a particular chromosome

Genotype: encoding of the population in the computation space.

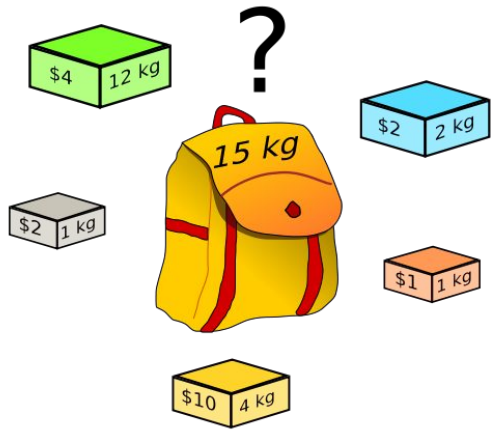
Phenotype: the population representation in the real world.

Genetic Algorithm: Problem Example

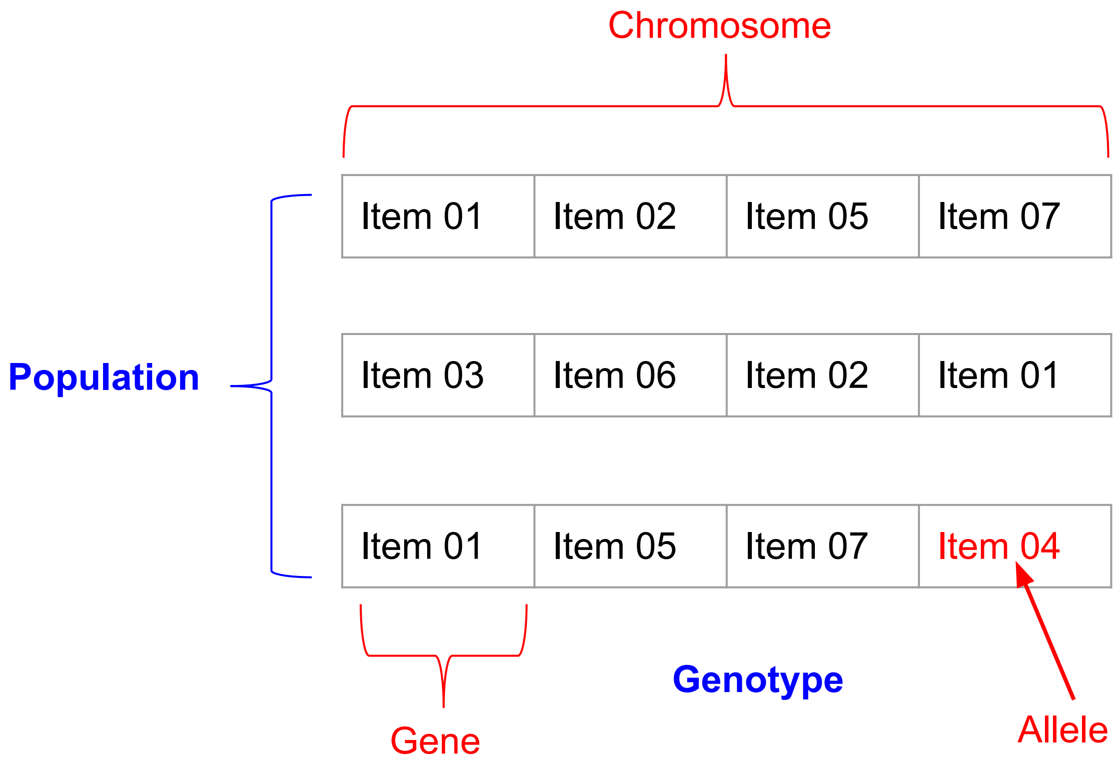
Given weights and values of ***n*** items, put these items in a knapsack of capacity ***W*** to get the maximum total value in the knapsack.



Genetic Algorithm: Encoding Example



Phenotype



Genetic Algorithm: Pseudo Code

```
 ''' General Genetic Algorith '''
Let t = 0 be the generation counter
Create and initialize an initial population P consists of n individuals

while stopping condition(s) not True:
    Evaluate the fitness of each individual 'i' in 'P'
    Select two or mor individual 'i' in 'P' to create offsprings
    Apply crossover over the selected indvidual to create "offspring"
    Apply mutation over the generated "offspring"
    Select Surivor individuals
    Advance to the new generation
    t = t + 1
end

return best indviduals in the last generation
```

Genetic Algorithm: Population Initialization

In general there are two methods:

Random Initialization: the initial population consist of completely random solutions.

Heuristic Initialization: use domain knowledge about the problem to define heuristic that we can use to to generate the initial population

Which one we should use and why?

Should we combine the two methods?

Genetic Algorithm: Population Initialization

Diversity of the individuals is the key to design a good population and build a better GA model.

There are two common population models, that identify how the population change from one generation to the next one

Incremental GA (Steady State): few offsprings are generated and replace the same number of parents from the population

Generational Model: n number of offsprings are generated where n is the population size and the entire population is replaced with the new one.

Genetic Algorithm: Population Initialization

What is Elitism?

- Is the process of ensuring that the best individuals of the current population survive to the next generation
- The best individuals are copied to the new population without being mutated.

Why Elitism is important?

What are the pros and cons of Elitism?

Genetic Algorithm: Population Survivor Selection

How do we select the individuals that will survive to the next generation?

- Age based Selection: each individual is allowed in the population for a finite number of generations regardless of its fitness.
- Fitness based Selection: the offspring replace the least fit individuals in the population

What is the size of age based selection in generational model?

What is the idea behind age based selection?

Genetic Algorithm: Encoding

How to map or model the phenotype in the genotype space?

Encoding is an important aspect when designing a genetic algorithm.

Bad or poor encoding usually leads to a GA with poor performance.

Four main encoding (representation) methods:

- Binary Encoding
- Value Encoding
- Permutation Encoding
- Tree Encoding

Genetic Algorithm: Binary Encoding

The simplest and the most widely used representation.

Very suitable for problems when the solution space consists of boolean decisions.

Example is the Knapsack problem:

- We have n items we use n bits string to represent the solution
- 1 means the item is selected and 0 means the item is not selected.

Chromosome 1 0 1 1 1 1 0 1 0 0 1 0

Genetic Algorithm: Value Encoding

For some problem the direct value encoding is the most suitable option

Using binary representation would be very difficult and problematic.

Some problem the solution consist of continuous values, for example finding the suitable weights for a neural network

What is the disadvantage of this encoding approach?

Chromosome	Up Left Left Up Up Down Right Up Left
Chromosome	1.38 0.57 1.09 2.19 5.14 1.46 0.39

Genetic Algorithm: Permutation Encoding

This type of encoding is only useful to encode the solution of ordering problems.

The chromosome in this case is a string of numbers (mostly integrates) that represent a specific sequence.

The order of the allele (values) of the genes in every chromosome is important.

Example is the travel salesman problem

Chromosome 5 3 7 8 1 4 2 6

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

Next-Time

Genetic Operators

Solving Problems with Genetic Algorithm

Genetic Algorithm Implementation

Questions