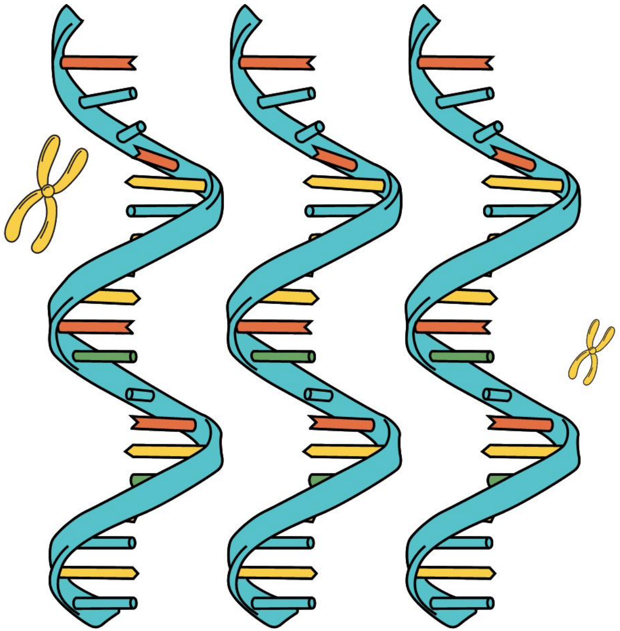
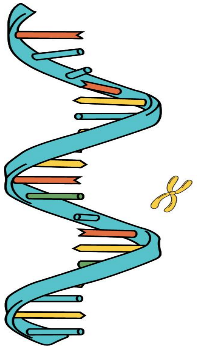


GAs

Introduction to Genetic Algorithm

From Theory to Practise

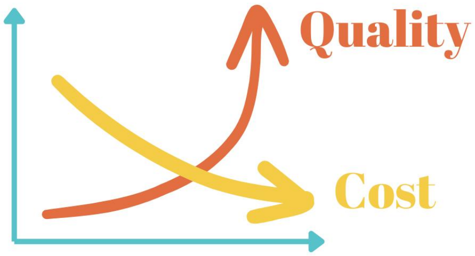
25 janvier 2024



CONTENTS

"The advance of genetic engineering makes it quite conceivable that we will begin to design our own evolutionary progress"

Isaac Asimov



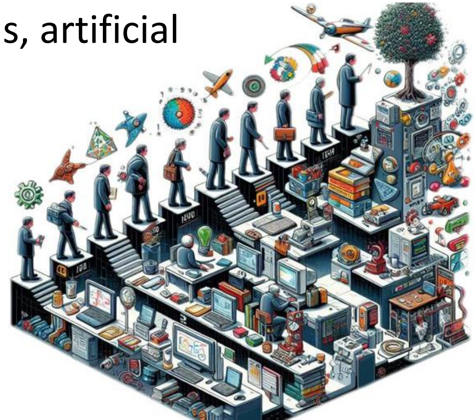
Introduction to Optimization

"True optimization is the revolutionary contribution of modern research to decision processes"

George Dantzig

Evolution of Problem-Solving

- Humans have grappled with problem-solving since ancient times.
- Today, we seek automated solutions for real life problems.
- Automated approaches rely on algorithmic solutions crafted for each problem's nature.
- There are many techniques for problem solving, such as numerical analysis, artificial intelligence, operational research, optimization, and more.



What is Optimization ?

- Optimization is derived from evolutionary algorithms and computational intelligence.
- An Optimization Problem is identified when the goal is to maximize or minimize a function.
- Maximization and Minimization problems can be easily converted to each other.

$$\min_x f(x) \iff \max_x[-f(x)]$$
$$\max_x f(x) \iff \min_x[-f(x)].$$

What is Optimization ?

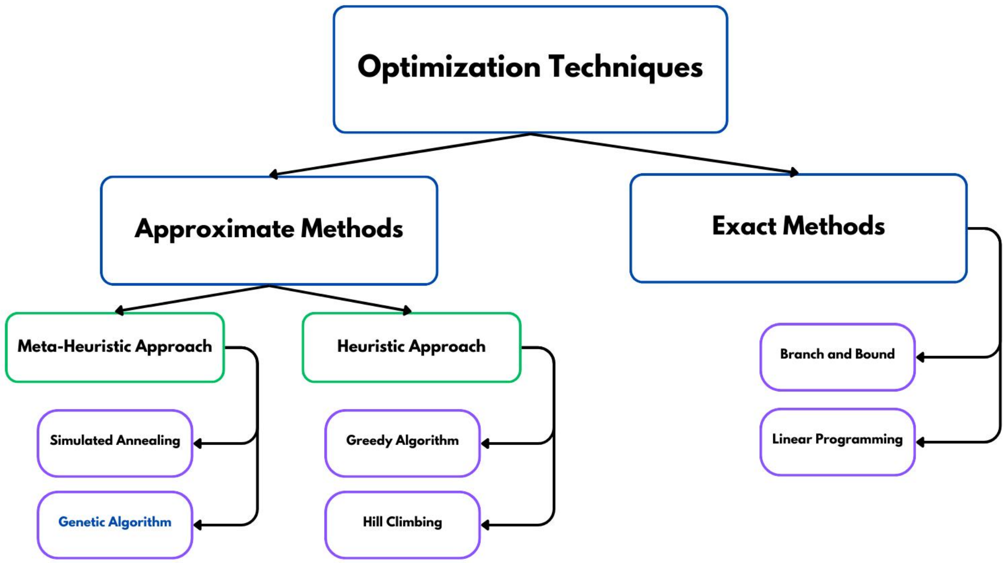
- The function being maximized or minimized is termed the objective function, denoted as f(x).
- The variable x is considered the independent variable or the solution feature.
- The objective function serves as an evaluative measure, determining if a given x is a favorable solution.
- Optimization involves finding the optimum solution within a population of potential solutions.

Optimization Approaches

Exact Methods: aim to find the globally optimal solution by exhaustively exploring the entire solution space.

Heuristic Methods: are problem-solving strategies that prioritize speed and practicality over guaranteed optimality.

Metaheuristic Methods: are higher-level strategies that guide other heuristics to explore the solution space efficiently.



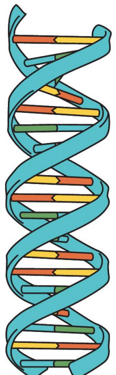
Understanding Genetic Algorithms

“The genetic algorithm is a model for learning and forms the heart of the field of artificial life.”

Melanie Mitchell

What is Genetic Algorithm ?

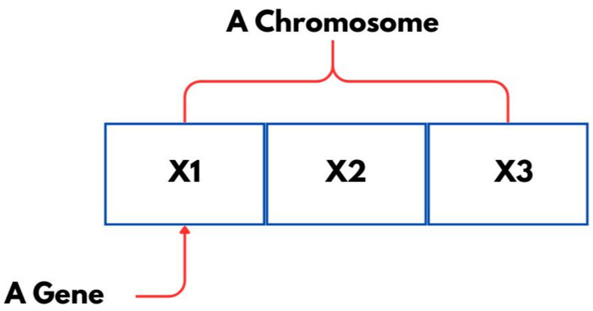
- Genetic Algorithms (GAs) are the earliest, most well-known, and widely-used Evolutionary Algorithms (EAs).
- GAs are highly effective tools for optimization.
- These algorithms draw inspiration from the process of natural selection observed in genes.



Analogy of Gentic Algorithm

- If you can represent each possible solution to the problem as a bit of string then GA may solve it for you.

Example : Consider you want to maximize the function $4x_1 - 2x_2 + 5x_3$ where $x_1, x_{2,} x_3 \in [0,3]$



Preparing for Implementation

Before diving into the implementation of the Genetic Algorithm, it's crucial to make two key decisions.

1. ***Decide Chromosome Encoding:*** **How do I want my solution to be presented?**

Considerations:

- Binary Encoding.
- Integer Encoding.
- Real Valued Encoding.
- Permutation Encoding.

1	0	1	0

2	7	5	4

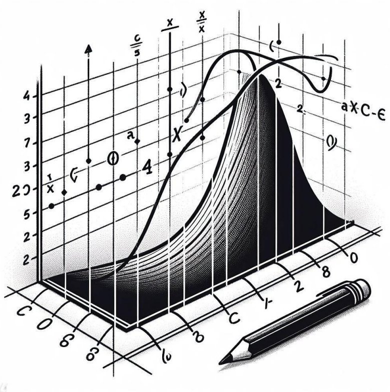
3.5	2.8	14.7	33.21

Preparing for Implementation

2. Determine Fitness Evaluation: **How do I know if this is a good solution or a bad one?**

Considerations:

- Define a fitness function that quantifies the quality of a solution.
- The function guides the algorithm toward optimal solutions.
- If the fitness function is not carefully chosen, reaching optimal solutions becomes
- challenging.



The Algorithm Process

- **Create the Individual Representaion:** Generate individual solutions representing potential solutions to the problem.

1	0	1	0

- **Population Initialization: -** Create a population by randomly generating multiple individuals.
  - Emphasize that the population size is an input parameter.

Indv1	1	1	1	0	0	0	1	0	Indv3
Indv2	0	0	1	1	1	1	1	1	Indv4

The Algorithm Process

- **Define a Fitness Function**

Example: f(x)= Sum(items)

- **Determine fitness for the population**

fitness(indv1)=3.
fitness(indv2)=2.
fitness(indv3)=1.
fitness(indv4)=4.

The Algorithm Process

Until Convergence or Number of Genearations = Max Repeat

- **Selection Operation**

After evaluating fitness for the entire population, choose individuals with the best fitness scores.

Common methods include tournament selection, roulette wheel selection, or rank-based selection.

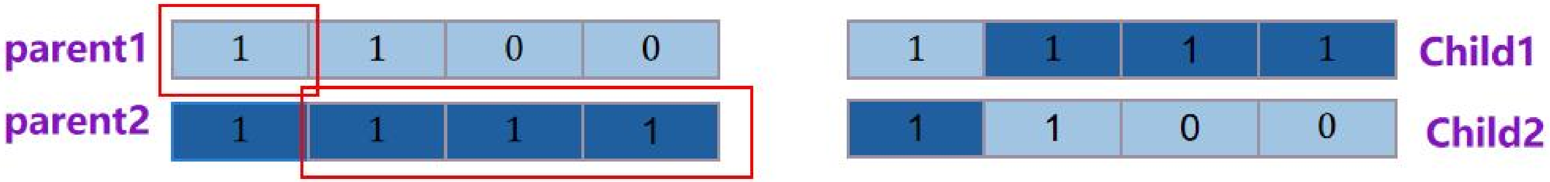
Selected individuals become parents for the next steps.

The Algorithm Process

- **Crossover Operation (**Mate Individuals**)**

Create pairs of parents from the selected individuals.

Randomly choose a crossover point.



The Algorithm Process

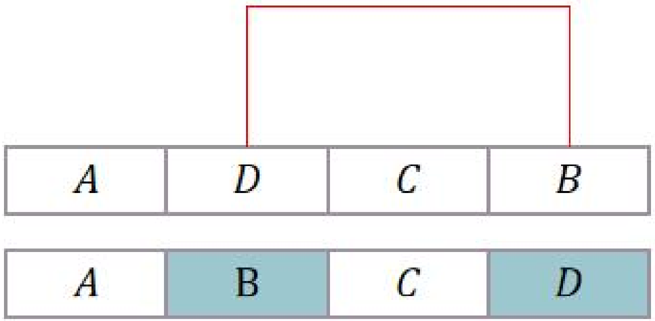
- **Mutation Operation**

Mutation involves making small random changes to individual genes.

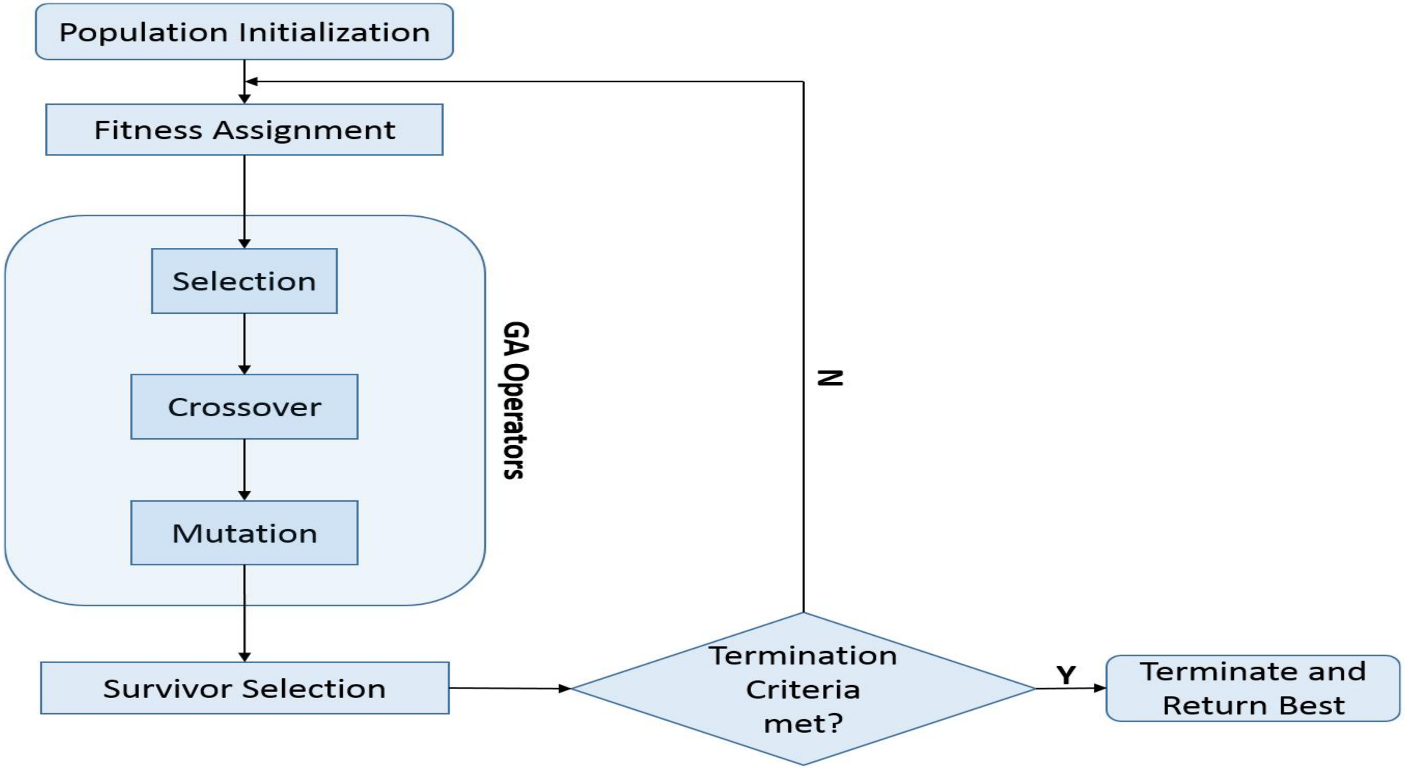
We Select a mutation Probability p

For each gene in the chromosome:

- Generate a random number r between 0 and 1.
- If $r \leq p$, mutate the gene; otherwise, leave it unchanged.



The Algorithm Process



Solving the Knapsack Problem

"The genetic algorithm is a model for learning and forms the heart of the field of artificial life."

Melanie Mitchell

What is the Knapsack Problem ?

Problem Definition:

- We are given an instance of the knapsack problem with an item set N, consisting of items represented by j with profit (Pj) and weight (Wj).

- The capacity value c. (Usually, all these values are taken from positive integer numbers.)

- The objective is to select a subset of items such that the total profit of the selected items is maximized, and the total weight does not exceed c.

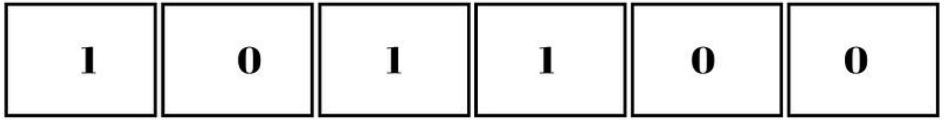
$$
\begin{array}{lll}
\text{(KP)} & \text{maximize} & \displaystyle\sum_{j=1}^{n} p_j x_j \\
 & \text{subject to} & \displaystyle\sum_{j=1}^{n} w_j x_j \leq c, \\
 & & x_j \in \{0,1\}, \quad j = 1, \ldots, n.
\end{array}
$$

Implementation of KP with a GA using Python

Individual Representation:

- the most effective representation for an individual is a binary format.
- We create a boolean array with a length equal to the number of items.
- Each cell is assigned either 1 or 0. A value of 1 indicates placing the item in the knapsack, while 0 signifies exclusion.

Array of length 6 representing 6 items: 0 for exclusion, 1 for inclusion



Implementation of KP with a GA using Python

We begin by creating our ‘Item’ class, where each item is characterized by a name, weight, and value (profit).

```python
# item.py

class Item:
    def __init__(self, name, value, weight):
        self.name = name
        self.value = value
        self.weight = weight

    def __repr__(self):
        return f"Item({self.name}, {self.value}, {self.weight})"
```

Implementation of KP with a GA using Python

- We define our ‘Knapsack’ class, where a knapsack problem is characterized by its capacity and a list of items.

```
# knapsack.py

class Knapsack:
    def __init__(self, capacity, items):
        self.capacity = capacity
        self.items = items
```

- The fitness function for our problem, involves of summing the values of the items considering the KP capacity.

```
def fitness(self. solution):
    total_value = 0
    total_weight = 0

    for i in range(len(solution)):
        if solution[i] == 1:
            total_value += self.items[i].value
            total_weight += self.items[i].weight

    # Check if the solution violates the capacity constraint
    if total_weight > self.capacity:
        # Penalize solutions that exceed the capacity
        return 0

    return total_value
```

Implementation of KP with a GA using Python

- The Selection Function employs a tournament-style selection, where individuals (potential parents) compete in groups of two.

- The winner of each tournament, determined by their fitness , becomes a selected parent.

```
def selection(self):
        tournament_size = 2
        selected_parents = []

        for _ in range(self.population_size):
            tournament = random.sample(self.population, tournament_size)
            selected_parents.append(max(tournament, key=lambda x: self.knapsack.

        return selected_parents
```

Implementation of KP with a GA using Python

- The Crossover Function generate a random probability.
- A crossover point is randomly chosen within the length of the parents.
- The genetic material beyond this point is exchanged between the parents, creating two new offspring child1 and child2.

```
def crossover(self, parent1, parent2):
    if random.random() < self.crossover_rate:
        crossover_point = random.randint(1, len(parent1) - 1)
        child1 = parent1[:crossover_point] + parent2[crossover_point:]
        child2 = parent2[:crossover_point] + parent1[crossover_point:]
        return child1, child2
    else:
        return parent1, parent2
```

Implementation of KP with a GA using Python

- For a given individual, the algorithm first creates a copy.
- For each element in this copy, there’s a check based on the mutation rate.
- If a randomly generated number is less than the mutation rate, the corresponding element is flipped (from 0 to 1 or vice versa).

```python
def mutation(self, individual):
    mutated_individual = individual.copy()
    for i in range(len(mutated_individual)):
        if random.random() < self.mutation_rate:
            mutated_individual[i] = 1 - mutated_individual[i]
    return mutated_individual
```

The GA Algorithm Paramters

- **Population Size:** Decides how many solutions to consider. More options mean more chances to find the best one.
- **Number of Generations:** Determines how many rounds of attempts. More rounds mean more chances for improvement.
- **Crossover Rate:** Controls how often solutions mix strategies. Higher rate means more mixing.
- Mutation Rate: Decides how often to make small, random changes. Keeps things diverse and avoids sticking to one idea.

Applications in the Tech World

“The genetic algorithm is a model for learning and forms the heart of the field of artificial life.”

Melanie Mitchell

Applications of Genetic Algorithm

NLP Processing

Feature selection, text classification model optimization, language model hyperparameter tuning, word embedding enhancement, grammar evolution

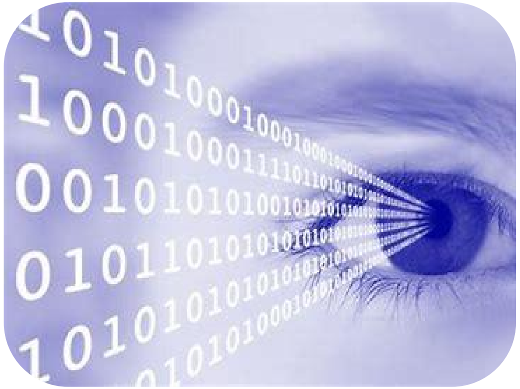


Image Processing

Image enhancement, feature selection, image segmentation, object recognition, and optimization of image processing parameters

Social Science

optimizing survey designs, evolving social network structures, optimizing decision-making strategies in social simulations

Loubna Bouzenzen

- Data Science and Big Data Analytics Enthusiast
- UX/UI Designer
- Flutter Developer
- Author
- Annaba, Algeria

@loubna.bouzenzen2112

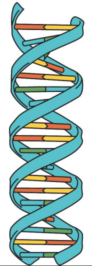
@loubna-bouzenzen

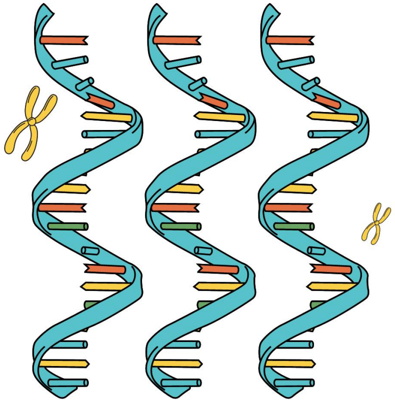
Detailled Medium Articles about GAs

- https://link.medium.com/7QPlqPjZvGb
- https://link.medium.com/b1yPbSkPCGb

Full Source code

- https://lnkd.in/dRb6zn66





THANK YOU

By Loubna Bouzenzen

Questions ?

Feel free to ask any questions or share your thoughts!