

National University of Computer and Emerging Sciences



Lab Manual 13
Artificial Intelligence Lab




Instructor: Mariam Nasim



Task 1

Implement Genetic Algorithm to solve the 0/1 Knapsack problem using the following configurations:

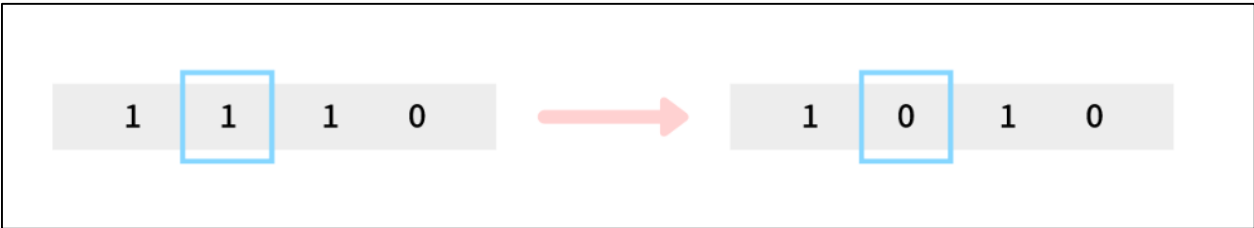
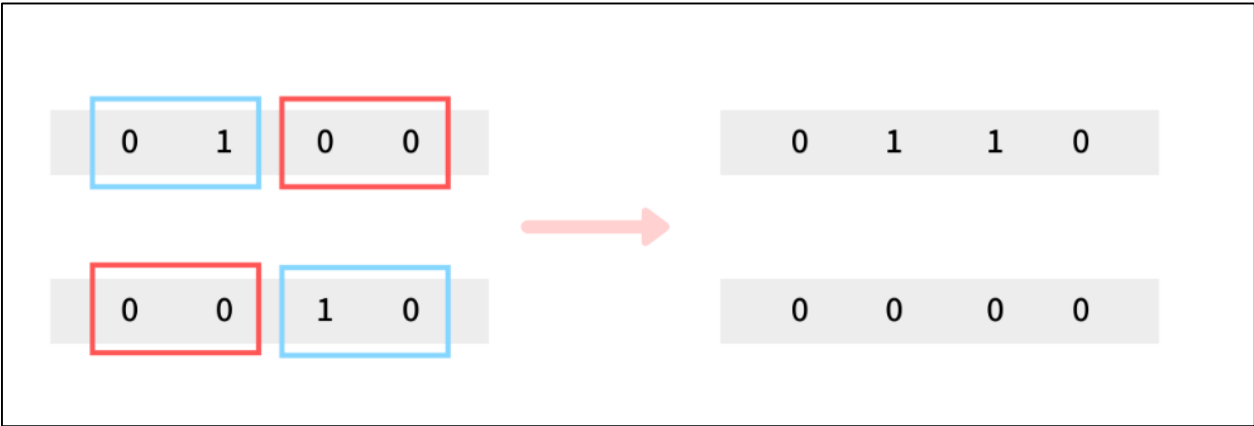
You are given a list of items, each with a weight and a value. Your task is to select a subset of these items to include in a knapsack such that the total weight does not exceed the given capacity, and the total value is maximized.

1. **Population Size:** 20
2. **Generations:** 50
3. **Selection Methods:**
 - *Linear Ranking Selection* (apply this first)
 - *Tournament Selection* (try this as a variant)
4. **Crossover:** *Uniform Crossover*
5. **Mutation:** *Bit Flip Mutation*
6. **Elitism:** Retain top 40% of the population in each generation

| | | | | | |
|---|---|---|---|---|---|
|  | 7 | 2 | 1 | 9 |  Max Weight: 15kg |
|  | 5 | 4 | 7 | 2 | |
| | A | B | C | D | |

| | | | | |
|---|----------|--------------------|----------|---|
|  | 7 | 2 | 1 | 9 |
|  | 5 | 4 | 7 | 2 |
| | A | B | C | D |
| | 1 | 0 | 0 | 1 |
| | A picked | B and C not picked | C picked | |

| | | | |
|---------------------------------------|---|---|---|
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| Randomly selected initial population. | | | |



Task 2A: Implement Minimax Without Alpha-Beta Pruning

Objective:

- Implement the Minimax algorithm to determine the optimal move for the AI without using alpha-beta pruning.
- Track the number of nodes evaluated by the Minimax algorithm

Steps:

1. **Game Setup:**
 - Represent the game state as the current count.
 - Each player can choose to add 1, 2, or 3 to the count.
2. **Minimax Implementation:**
 - Implement the basic Minimax algorithm without pruning.
 - The evaluation function should return:
 - +1 for a win for the AI.
 - -1 for a win for the opponent.
 - 0 for a draw.
3. **Game Play:**
 - Allow the player to make a move by entering a number (1, 2, or 3).
 - The AI will choose the optimal move using Minimax.
 - Display the current count and indicate the winner.

Task 2B: Implement Minimax With Alpha-Beta Pruning

Objective:

- Optimize the Minimax algorithm using **Alpha-Beta Pruning** to reduce the number of nodes evaluated.

Steps:

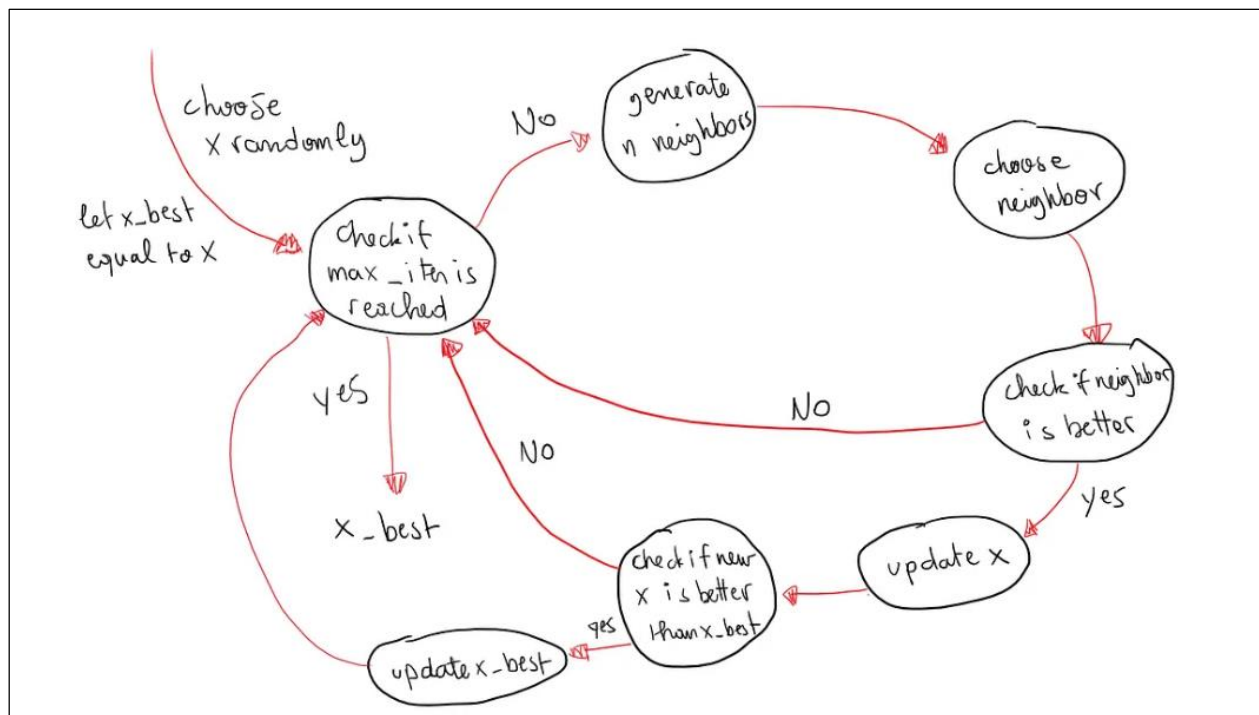
1. **Review Minimax Implementation:**
 - Reuse the Minimax function from **Task 4A**.
 - Introduce the alpha and beta parameters to prune unnecessary nodes.
2. **Alpha-Beta Pruning Implementation:**
 - Modify the Minimax function to include alpha and beta parameters.
 - Track the number of nodes evaluated after applying pruning.
3. **Comparison and Analysis:**
 - Run both implementations on the same game states.
 - Compare the number of nodes evaluated with and without pruning.
 - Analyze how pruning affects the AI's decision-making speed.

Task 3: Traveling Salesman problem using Hill climbing

The Traveling Salesman Problem (TSP) is a well-known combinatorial optimization problem that has been extensively studied in operations research and computer science. The objective is to determine the shortest possible route that visits each city once and returns to the origin city. Due to its NP-hard nature, exact solutions are computationally expensive for large datasets, making heuristic and metaheuristic approaches like Hill Climbing.

For hill climbing, starting from an initial solution, it explores its immediate neighborhood, making small modifications or steps to the current solution. If a neighboring solution is better (according to the problem's objective function), the algorithm moves to that solution. This process repeats until a local maximum is reached, meaning that no further improvements can be made by moving to adjacent solutions.

Example flow of implementing Travelling salesman problem using hill climbing:



Implement traveling salesman problem using hillclimbing.