Documentație Algoritm evolutiv pentru Problema Rucsacului

1. Description and Pseudocode

A genetic algorithm is an adaptive heuristic search algorithm that simulates the process of natural evolution. For the Knapsack Problem, it is used to find an approximate solution to the combinatorial problem of maximizing the value of items placed in a knapsack without exceeding its capacity.

Pseudocode:

```
Initialize population with random individuals
Evaluate the fitness of each individual in the population
for each generation until termination criteria are met
    Select parents from the population
    Perform crossover to produce offspring
    Apply mutation to the offspring
    Evaluate the fitness of the new offspring
    Select individuals for the next generation from current and offspring
end for
Return the best individual as the solution
```

2. Components

- **Solution Representation:** Solutions are represented as binary strings where each bit represents the inclusion or exclusion of an item in the knapsack.
- **Fitness Function:** Calculates the total value of items in the knapsack and penalizes solutions that exceed the capacity.
- **Selection Operator:** Tournament selection is used to choose parent individuals for reproduction.
- **Crossover Operator:** A one-point or uniform crossover is used to combine two parents into a new offspring.
- **Mutation Operator:** Bit-flipping mutation is used to introduce variability.
- **Algorithm Parameters:** Population size, mutation rate, crossover rate, and the number of generations.

3. Algorithm Parameters

- Population Size: Defines the number of individuals in the population.
- Mutation Rate: The probability that an allele (bit) will be mutated.
- Crossover Rate: The probability that two individuals will be crossed to produce offspring.
- Number of Generations: The number of cycles the algorithm will run before termination.

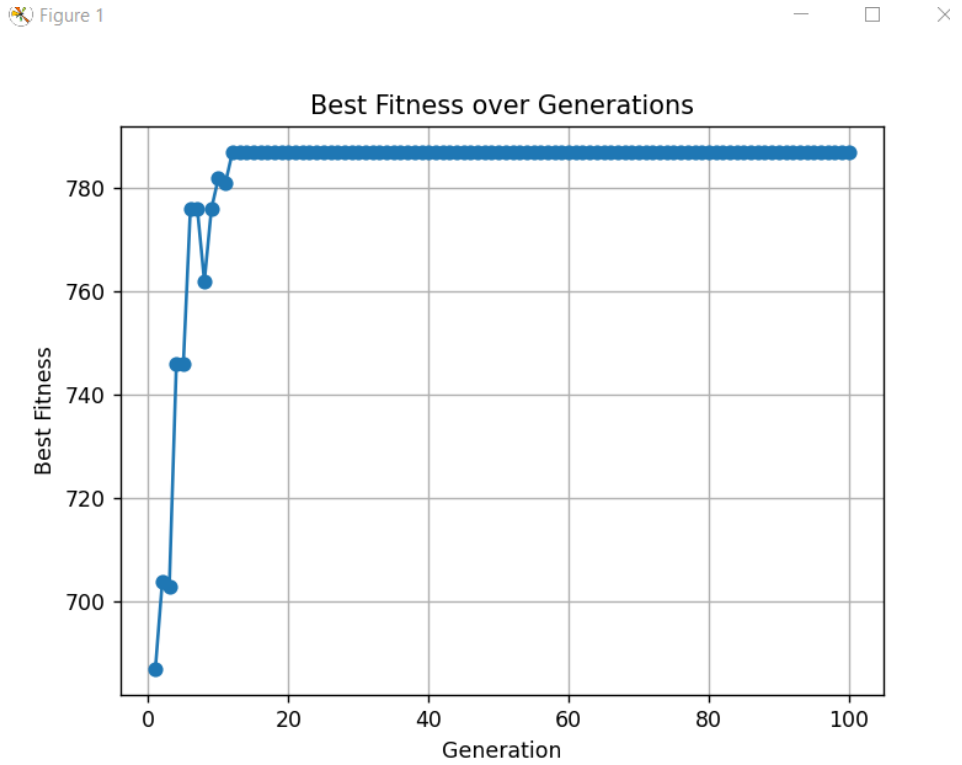
4. Results

The algorithm was applied to two instances of the knapsack problem with different numbers of items: a smaller instance with 20 items and a larger one with 200 items. The capacity of the knapsack for each test was set based on the problem instance.

Comparison with Previous Algorithms (Random Search, NAHC)

Parameter Set	**Population Size**	**Mutation Rate**	**Crossover Rate**	Tournament Size	**Best Value (20 items)**	**Avg. Value (20 items)**	**Best Value (200 items)**
Conservative Strategy	100	0.01	0.9	2	782	778.99	95701

Parameter Set	Population Size	Mutation Rate	Crossover Rate	Tournament Size	Best Value (20 items)	Avg. Value (20 items)	Best Value (200 items)
Explorative Strategy	100	0.02	0.8	5	787	775.78	97024
Balanced Strategy	50	0.01	0.95	3	772	764.39	96346.00
Intensive Crossover	200	0.015	0.85	7	787	781.43	97469



- **Conservative Strategy**: A parameter set with a smaller population and lower rates, focusing on slow, steady exploration of the search space.
- **Explorative Strategy**: A set with a higher mutation rate, encouraging diverse exploration at the risk of disrupting good solutions.
- **Balanced Strategy**: A configuration that aims to maintain a balance between exploration and exploitation with moderate rates.
- **Intensive Crossover**: This set features a high crossover rate, focusing on combining existing solutions to find better ones.

3.Analysis

The genetic algorithm (GA) was applied to the knapsack problem using four different parameter sets, each representing a unique strategy. The performance of these strategies was measured against two problem instances: one with 20 items and another with 200 items. Here, we analyze the results obtained from these experiments.

Observations

- **Overall Performance**: All strategies managed to find high-value solutions for both the 20 and 200 item problems, demonstrating the effectiveness of GA in solving combinatorial optimization problems.

- **Best Value Achievement**: The **Intensive Crossover** strategy achieved the highest best value for the 200-item problem, suggesting that a high crossover rate combined with a large population size effectively explores and exploits the solution space.
- **Average Value Consistency**: The **Conservative Strategy** showed less variance in the average values for the 20-item problem, indicating its consistency and reliability in finding near-optimal solutions across runs.
- **Impact of Mutation Rate**: The **Explorative Strategy**, with a higher mutation rate, had slightly lower average values for the 200-item problem compared to other strategies. This might suggest that while a higher mutation rate encourages exploration, it could also disrupt potentially good solutions.
- **Population Size and Tournament Size**: The **Balanced Strategy** and **Intensive Crossover** strategy, which used different population sizes but more focused selection pressures (via tournament size), showed that a careful balance of population dynamics and selection pressure is crucial for the GA's success.

4.Conclusions

The experiments indicate that the genetic algorithm is a robust method for tackling the knapsack problem, capable of adapting to different problem scales through parameter tuning. The **Intensive Crossover** strategy proved to be particularly effective for the larger problem size, highlighting the importance of crossover in generating quality solutions. However, the **Conservative Strategy**'s consistency for the smaller problem set underscores the value of a more restrained approach in certain contexts.