**Solving Knapsack Problem Using Genetic Algorithm**

**By**

**Mownika Konamaneni**

**Bindhu Lasya Nandimandalam**

**Siri Kesidi**

# **Project Description:**

This project aims to leverage the power of Genetic Algorithms to tackle the Knapsack Problem, a classical optimization challenge. In this context, an individual equipped with a backpack enters a store where an array of 10 items is available, each characterized by its unique weight and price. The primary objective is to strategically curate a selection of items that maximize the total value within the confined weight limits of the knapsack.

The primary challenge revolves around the meticulous selection of items, maximizing the total value within the constraints of the knapsack's weight capacity, set at 45 kg. The Genetic Algorithm serves as the key tool for refining the selection strategy, employing an iterative and evolutionary approach. Through successive generations, the algorithm fine-tunes the composition of the knapsack, eventually converging to an optimized solution that achieves the highest possible value without breaching the weight limit.

Applying the theory of genetic algorithms, the problem is approached through the formation of a population using a random-based search technique. Genetic algorithms are widely utilized in contemporary research due to their efficacy in addressing practical optimization challenges. The genetic algorithm operates by iteratively forming a population, introducing new solutions into the search space for evaluation, and potentially incorporating them into the population. The effectiveness of a genetic algorithm is contingent on the performance of key operators, including selection, crossover, and mutation.

Motivated by the potential for improvement in the performance of genetic algorithms, this project aims to explore the application of genetic algorithms to the Knapsack Problem. By investigating and enhancing the various stages of the genetic algorithm, such as population formation and operators, the study seeks to contribute to the optimization of solutions for the Knapsack Problem.

# **Genetic Algorithm Overview:**

Genetic Algorithm (GA) is an Evolutionary Algorithm that mimics natural evolution to find solutions for optimization problems. It operates with a population of candidate solutions represented as strings (chromosomes). Key concepts inspired by natural evolution include inheritance, mutation, selection, and crossover.

## **Algorithm Workflow:**

The GA initiates with an initial population of candidate solutions (chromosomes) and iteratively refines these solutions over generations. The workflow includes evaluating fitness, selecting individuals for reproduction, applying crossover and mutation, and replacing the old population with the new.

* + 1. **Initialization:**

The algorithm begins by creating an initial population of candidate solutions, known as chromosomes. Each chromosome represents a potential combination of items for the knapsack. The initial population is randomly generated, fostering diversity and exploration of a broad solution space.

* + 1. **Fitness Evaluation:**

The fitness of each chromosome is evaluated based on predefined criteria. For the 0–1 Knapsack problem, the fitness function assesses the total value of selected items while ensuring the knapsack's weight constraint is not violated.

* + 1. **Selection:**

The selection process determines which chromosomes are more likely to contribute to the next generation. Individuals with higher fitness scores have a greater chance of being selected. Various selection methods, such as roulette-wheel, tournament, or stochastic selection, can be employed.

* + 1. **Reproduction:**

Reproduction involves creating the next generation of chromosomes through genetic operations. Two fundamental genetic operators, crossover and mutation, are applied to generate offspring. Crossover combines genetic material from selected parents, while mutation introduces small, random changes to foster diversity.

* + 1. **Replacement:**

The new generation, comprising offspring and potentially some surviving individuals from the previous generation, replaces the old population. The replacement process is crucial for maintaining genetic diversity and driving the algorithm towards optimal solutions.

* + 1. **Termination Conditions:**

Termination conditions determine when the algorithm should stop iterating. Conditions may include reaching a predefined number of generations, achieving a satisfactory solution, or detecting convergence. Monitoring and assessing these conditions guide the algorithm towards convergence or termination.

The iterative interplay of these stages defines the GA workflow, providing a robust methodology for solving complex combinatorial optimization problems like the 0–1 Knapsack. This cyclic process continues until termination conditions are met, yielding a set of candidate solutions with progressively improved fitness over generations.

# **Application to 0–1 Knapsack Problem**

The application of the Genetic Algorithm (GA) to the 0–1 Knapsack problem involves a strategic adaptation of genetic principles to efficiently address this combinatorial optimization challenge. This section delves into the specific techniques and considerations applied when utilizing the GA for solving the 0–1 Knapsack problem.

|  |  |  |
| --- | --- | --- |
| Item No. | Weight | Value |
| 1 | 11 | 235 |
|  |  |  |
| 2 | 11 | 430 |
|  |  |  |
| 3 | 13 | 166 |
|  |  |  |
| 4 | 13 | 256 |
|  |  |  |
| 5 | 3 | 388 |
|  |  |  |
| 6 | 5 | 174 |
|  |  |  |
| 7 | 15 | 40 |
|  |  |  |
| 8 | 1 | 442 |
|  |  |  |
| 9 | 10 | 113 |
|  |  |  |
| 10 | 17 | 91 |

* + 1. **Representation of Items:**

The 0–1 Knapsack Problem revolves around selecting items with specific weights and values to maximize the overall value while adhering to the knapsack's weight capacity. Items can be represented using a 2D Array having two columns containing value and weight of items respectively. The table at the right lists the available items with their respective weights and values.

* + 1. **Representation of chromosomes:**

The foundation of the Genetic Algorithm's approach to the 0–1 Knapsack Problem lies in the representation of items. Each item is encoded as a binary bit in a chromosome, indicating whether the item is selected (1) or not (0). This binary representation allows for a concise and efficient exploration of potential solutions.

Chromosome example:

chromosome = [1, 0, 1, 1, 0]

*Representation of items 1*

The presence of 1s at the first, third and fourth positions indicates that items 1,3 and 4 are selected.

The absence of 0s at the second and fifth positions means that items 2 and 5 are not included in the knapsack.

* + 1. **Fitness Function:**

The fitness function for the 0–1 Knapsack problem is designed to evaluate the optimality of a solution. It considers both the total value of the selected items and adherence to the knapsack's weight constraint. The goal is to maximize the total value while ensuring the sum of the selected item weights does not exceed the knapsack's capacity.

* + 1. **Initialization Strategy:**

Our GA begins with a diverse population achieved through random initialization, laying the foundation for exploration and the discovery of varied item combinations.

Example:  
[1 1 0 1 1 1 1 1 1 0]

[0 1 0 0 1 1 1 0 1 0]

[1 0 0 0 0 1 0 0 1 0]

[0 0 0 1 0 0 1 1 0 0]

[1 0 1 0 1 0 1 1 0 1]

[0 1 0 1 0 0 1 0 0 1]

[1 0 0 0 1 1 1 0 0 0]

[0 0 0 0 0 1 1 1 0 0]

[0 1 1 1 1 0 0 0 1 0]

[0 0 1 1 0 0 1 0 0 1]

* + 1. **Selection Methods:**

Tournament selection is employed to choose potential parents for reproduction. This method promotes the selection of individuals with higher fitness, contributing to the diversity of the offspring.

* + 1. **Reproduction**

In this step the next generation of population is created using the following genetic operators:

1. **Crossover:**

Crossover simulates the genetic recombination observed in nature. For the knapsack problem, it entails exchanging genetic material between two parent individuals to create new offspring. The choice of crossover points and methods significantly influences the diversity and quality of the population. We set the crossover rate to 0.8.

1. **Mutation**:

Mutation introduces random changes to individuals to maintain genetic diversity. It simply means changing a bit from 0 to 1 or 1 to 0. Mutation is done to prevent some random loss of potentially useful solution due to reproduction or crossover. It is done with a very small probability of about one mutation per thousand bits of transfer. We set the mutation probability rate as 0.4.

1. **Termination**:

The algorithm continues these steps for a predefined number of generations or until a termination criterion is met, such as a satisfactory fitness level or a maximum runtime.

**Termination Criteria**:

The algorithm terminates under the following conditions:

* A satisfactory solution meeting minimum criteria is found.
* The maximum number of generations is reached.
* The allocated time for execution expires.

1. A graph of a graph with text

   Description automatically generated with medium confidence**Results**
   1. **Tournament Selection**

The Genetic Algorithm, with tournament selection as its cornerstone, exhibited noteworthy results in addressing the 0–1 Knapsack Problem. The algorithm initiated with an initial population of chromosomes, each representing a potential solution to the optimization problem.

* 1. **Generation Evolution**

The evolutionary process unfolded across multiple generations, refining candidate solutions iteratively. The fitness of each chromosome, a measure of its adherence to knapsack constraints and its ability to maximize value, played a pivotal role. Tournament selection, a crucial step in the reproductive phase, ensured that chromosomes with higher fitness values had an increased likelihood of being selected as parents.

1. Fitness Through The Genrations 1
   1. **Convergence and Optimal Solutions**

As the algorithm progressed, it successfully converged to a solution that not only satisfied the knapsack constraints but also maximized the overall value of the selected items. The final generation showcased chromosomes with commendable fitness values, indicating the algorithm's proficiency in navigating the solution space effectively.

# **Conclusion**

The results obtained from the application of the Genetic Algorithm, particularly when employing tournament selection, highlight its effectiveness in optimizing the selection process for the 0–1 Knapsack Problem. The adaptability and efficiency demonstrated in this study make Genetic Algorithms, with tournament selection, a valuable tool for resource allocation and constraint optimization scenarios.