**0-1 Knapsack using hybrid of PSO and Genetic Algorithm**

**Project Based Learning**

**M.Tech (CSE) 2022-24**

# 

# JAYPEE INSTITUE OF INFORMATION TECHNOLOGY

# SECTOR-62, NOIDA. (U.P)

**Submitted by- Subject Coordinator  
Sushrut Khajuria (MTEG220017) Dr. Parul Agarwal   
Prankur Rana (MTEG220034)**

**Section-I**

**INTRODUCTION**

Genetic Algorithm are one of the evolutionary algorithms that use techniques which are inspired from Darwin’s Theory. In this project, we address the 0-1Knapsack issue using genetic algorithms. The goal is to maximise the usefulness of the items in a knapsack without filling it to capacity. Approaches like dynamic programming, backtracking, branch and bound, etc. are not very helpful for solving the Knapsack problem because it is an NP problem. When it comes to finding solutions to issues like the Knapsack problem, which is typically regarded as computationally infeasible, genetic algorithms clearly outperform all other approaches.

**MOTIVATION**

The reason we opted for this project is because Genetic Algorithm are the heuristic search and optimization techniques that in reality mimic the process that we can observe in nature. As the Algorithm has been used widely for optimization problems these days and is also used in business, research and development and other engineering disciplines.

**ABOUT THE PROJECT**

This project outlines a study that employed Genetic Algorithms (GAs) to resolve the 0-1 Knapsack Problem (KP). An example of a combinatorial optimization issue is The Knapsack Problem, which aims to maximise the benefit of items in a knapsack without filling it to capacity.

**Section-II**

**GENETIC ALGORITHM**

Genetic Algorithms are computer algorithms that search for good solutions to a problem from among a large number of possible solutions. They were proposed and developed in the 1960s by John Holland, his students, and his colleagues at the University of Michigan. The principles of natural evolution, such as selection by characteristics such as fitness, reproduction, and mutation, served as the basis for these computational paradigms. These mechanics are excellently suited to address a wide range of real-world issues, including computational issues, in numerous disciplines. Optimization, automatic programming, machine learning, economics, immune systems, population genetics, and social systems are some areas in which GAs are used.

**Basic elements of Genetic Algorithm**

Most GAs methods are based on the following elements, populations of chromosomes, selection according to fitness, crossover to produce new offspring, and random mutation of new offspring.

**Chromosomes**

The chromosomes in GAs represent the space of candidate solutions. Possible chromosomes encodings are binary, permutation, value, and tree encodings. For the Knapsack problem, we use binary encoding, where every chromosome is a string of bits, 0 or 1

**Fitness Function**

GAs require a fitness function which allocates a score to each chromosome in the current population. Thus, it can calculate how well the solutions are coded and how well they solve the problem.

**Selection**

The selection process is based on fitness. Chromosomes that are evaluated with higher values (fitter) will most likely be selected to reproduce, whereas, those with low values will be discarded. The fittest chromosomes may be selected several times, however, the number of chromosomes selected to reproduce is equal to the population size, therefore, keeping the size constant for every generation. This phase has an element of randomness just like the survival of organisms in nature. The most used selection methods, are roulette-wheel, rank selection, steady-state selection, and some others.

**Crossover**

Crossover is the process of combining the bits of one chromosome with those of another. This is to create an offspring for the next generation that inherits traits of both parents. Crossover randomly chooses a locus and exchanges the subsequences before and after that locus between two chromosomes to create two offspring.

**Mutation**

Mutation is performed after crossover to prevent falling all solutions in the population into a local optimum of solved problem. Mutation changes the new offspring by flipping bits from 1 to 0 or from 0 to 1.

**Simple Genetic Algorithm Pseudo Code**

1. **Initiate the parameters**
2. **Initialize the population  
    a) Put values in Chromosomes**

**b) Create Chromosomes within search range**

**c) Compute fitness on population**

**3) Start the loop for iterations**

**4) Recombination**

**a) Selection  
 b) Crossover  
 c) Mutation**

**Knapsack Problem**

The Knapsack Problem is a combinatorial optimization Problem. Given a set of items with a weight and value, it seeks to select a number of items to be placed in a Knapsack of fixed capacity such that total weight of the items is less or equal to the capacity but its value is as large as possible. The problem often arises in resource allocation with financial constraints.

**Programming language used**

**Python-**

It is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming.

The latest released version of python is 3.10.7

And we are using the same latest version of python for optimization of knapsack problem.

Parameter settting

Datasets

Eval measures