STUDY OF EVOLUTIONARY ALGORITHMS TO SOLVE 0/1 KNAPSACK PROBLEM

Ritika Mahajan*

Sarvesh Chopra**

ABSTRACT

This paper presents evolutionary algorithms design paradigms applied to single problem – 0/1 Knapsack Problem. The Knapsack Problem is a combinatorial optimization problem where one has to maximize the benefits of objects in a knapsack without exceeding its capacity. An optimization problem is simply a problem for which there are different possible solutions, and there is some clear notion of solution quality. That is, an optimization problem exists when different candidate solutions can be meaningfully compared and contrasted. Optimization algorithm can be deterministic or probabilistic. Deterministic techniques such as Greedy algorithm, dynamic programming technique are not best suited for such problems. Our objective is to study that how the evolutionary techniques - Genetic and Memetic Algorithms affect the performance of Knapsack Problem. A population-based search algorithm called Genetic Algorithm (GA) is commonly used to solve combinatorial optimisation problems where the goal is to find the best solution in a (possibly unknown) solution space. It uses the principle of biological evolution to generate successively better solutions from previous generations of solutions. Memetic algorithm (MA) is an extension of GA which incorporates a local-search algorithm for each solution in between generations.

Keywords: *Knapsack Problem, Genetic Algorithm, Memetic algorithm.*

*Shaheed Bhagat Singh College of Engineering and Technology, Ferozepur, India.

**Guru Nanak Dev Engineering College, Ludhiana, India.

1. INTRODUCTION

The Knapsack problem is a combinatorial optimization problem where one has to maximize the benefit of objects in a knapsack without exceeding its capacity. Given a set of items we have to find optimal packing of a knapsack. Each item is characterized by weight and value and knapsack is characterized by capacity. Optimal packing is the one in which weight is less or equal to the capacity and in which value is maximal among other feasible packing.

More formally:

Given a number of items n, their weights W = w1……wn,

their values V = v1……vn and knapsack capacity c,

find vector X = x1…..xn so that (x1*w1 + …..xn*wn) <=c

and (x1*w1 + …..xn*wn) is maximal. [1]

Knapsack Problem is a NP-complete problem. NP-complete problem is that no fast solution to them is known. Although any given solution to such a problem can be verified quickly, there is no known efficient way to locate a solution in the first place. The time required to solve the problem using any currently known algorithm increases very quickly as the size of the problem grows. As a result, the time required to solve even moderately large versions of many of these problems easily reaches into the billions or trillions of years, using any amount of computing power available today. A problem is said to be *tractable* if it can be solved by an algorithm in polynomial time with respect to the "size" of the problem. [2]

1.1 Applications

- Knapsack problem is the basis for a public key encryption system. [3]
- One early application of knapsack algorithms was in the construction and scoring of tests in which the test-takers have a choice as to which questions they answer. On tests with a homogeneous distribution of point values for each question, it is a fairly simple process to provide the test-takers with such a choice.[4]
- For example, if an exam contains 12 questions each worth 10 points, the test-taker need only answer 10 questions to achieve a maximum possible score of 100 points. However, on tests with a heterogeneous distribution of point values—that is, when different questions or sections are worth different amounts of points— it is more difficult to provide choices. Feuerman and Weiss proposed a system in which students are given a heterogeneous test with a total of 125 possible points. The students are asked to answer all of the questions to the best of their abilities. Of the possible

subsets of problems whose total point values add up to 100, a knapsack algorithm would determine which subset gives each student the highest possible score.[4]

2. GENETIC ALGORITHM

A genetic algorithm (GA) is a search heuristic that mimics the process of natural evolution. This heuristic is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.

Genetic Algorithms are search algorithms based on natural selection and genetics. This idea was first developed by John H. Holland in the early 1970's. The algorithm combines a random selection by the survival of the fittest theory. We can say that the strongest individuals in a population will have a better chance to transfer their genes to the next generation. Simply we can code a number of different solutions of a problem as a bit string, and evaluate their fitness in relation to each other. Every solution can be seen as an individual in a population, and the bit string can be seen as the genes of the individual. We then select the individuals to reproduce by weighted Monte Carlo selection based on their fitness. [5]

The reproduction can be done in three ways:

Pure Reproduction - The individual is copied directly into the next generation (cloning).

Crossover - Two individuals are selected and their genes are crossed at some point, as the first part of the new individual comes from one parent and the last part from the other.

Mutation - An individual is selected, and one bit is changed. Reproduction will be done until we have the chosen number in each generation. When this is done for several generations, we will hopefully have found an optimal solution, or a solution which is close to the optimum.

A typical genetic algorithm requires:

1. a genetic representation of the solution domain,
2. a fitness function to evaluate the solution domain.

Genetic algorithms find application in bioinformatics, phylogenetics, computational science, engineering, economics, chemistry, manufacturing, mathematics, physics and other fields. [6]

2.1 GA Process

1. Randomly generate initial population
2. Until a solution is found:
    1. Score each individual in the population
    2. Randomly select individuals as survivors

3. Perform crossover on survivors
4. Perform mutation on new generation (small chance)

3. Report the best solution

3. MEMETIC ALGORITHM

A population-based search algorithm called Genetic Algorithm (GA) is commonly used to solve combinatorial optimisation problems where the goal is to find the best solution in a (possibly unknown) solution space. It uses the principle of biological evolution to generate successively better solutions from previous generations of solutions. Memetic algorithm (MA) is an extension of GA which incorporates a local-search algorithm for each solution in between generations. According to Pastorino (2004), MA is able to improve convergence time, hence making it more favourable over GA. [7]

Local search is performed in between each generation, in addition to the techniques used by GA to explore the search space, namely recombination/crossover and mutation. For this reason, Memetic Algorithm is also known as Hybrid-GA (Moscato 2002). Local search is performed to improve the fitness of the population (in a localised region of the solution space) so that the next generation has "better" genes from its parents, hence the claim that Memetic Algorithms can reduce convergence time. Memetic Algorithms incorporate the concept of memes by allowing individuals to "change" before the next population is produced. Individuals may "copy" parts of genes from other individuals to improve their own fitness.

The local search algorithm adopted in a Memetic Algorithm is somewhat dependent on the problem being solved; however the common trait with any local search is that parameters in the algorithm cannot be changed. This does not follow with the definition of a meme, in that it can be changed because the adopted meme is the individual's own interpretation of it. Furthermore, when memes are transmitted, changes to them are also passed on. Memes affect the behaviour of an individual, and do not modify the genes themselves. However, as a practical issue, a meme in a Memetic Algorithm must be able to modify genes in order to improve fitness during local search. [8]

MAs include a broad class of metaheuristics. This method is based on a population of agents and proved to be of practical success in a variety of problem domains. We can be sure that MAs constitute one of the most successful approaches for combinatorial optimization in general, and for the approximate solution of NP Optimization problems in particular. Unlike traditional Evolutionary Computation approaches, MAs are concerned with exploiting all

available knowledge about the problem under study. This is not as an optional mechanism, but as a fundamental feature. [9]

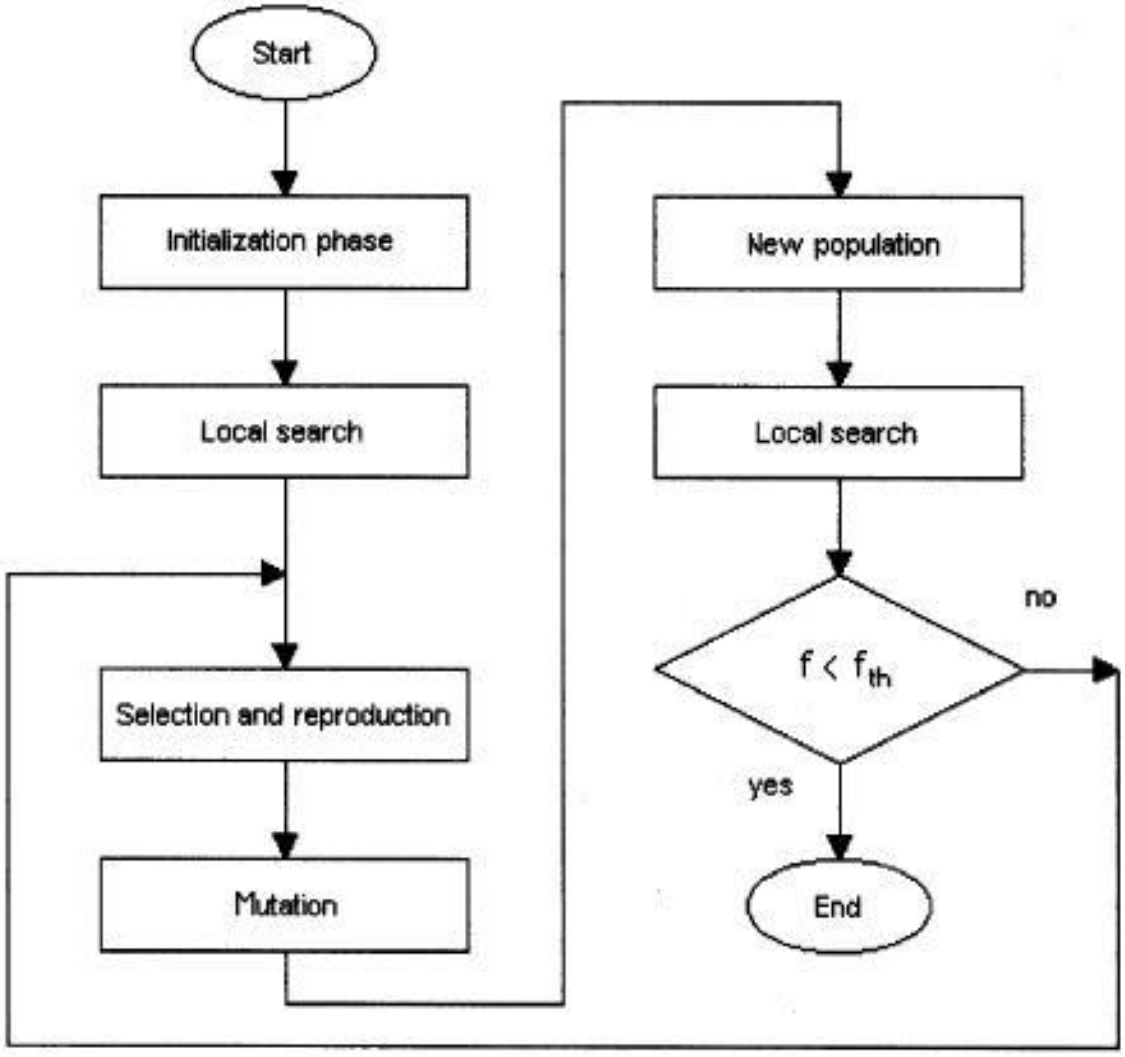

Fig. 1. Scheme of the memetic algorithm. f_{th} denotes a fixed threshold value used for the stopping criterion.

3.1 Applications Of Memetic Algorithms

The numerous applications of MAs are-

- Traditional *NP* Optimization problems
- Combinatorial Optimization Problems
- Machine Learning and Robotics
- Electronics and Engineering
- Molecular Optimization Problems

MAs have been also utilized in other fields such as, for example, *medicine, economics, oceanography, mathematics, imaging science and speech processing* etc. [10]

4. CONCLUSION

Two Evolutionary Algorithm techniques, that is, Genetic Algorithm and Memetic Algorithm are discussed. Deterministic techniques such as Greedy algorithm, dynamic programming technique are not best suited for NP problems. GAs reduces the complexity of the Knapsack Problem which makes it possible to find approximately optimal solutions for NP problem.

5. REFERENCES

1. Knapsack problem <http://www.utdallas.edu/~scniu/OPRE-6201/documents/DP3-Knapsack.pdf>
2. NP-complete. [Online]. Available: http://en.wikipedia.org/wiki/NP-complete
3. NP-hard-Wikipedia <http://en.wikipedia.org/wiki/NP-hard>
4. Knapsack problem <http://en.wikipedia.org/wiki/Knapsack_problem>
5. Arild Hoff, Ingvar Mittet and Arne Løkketangen, Genetic Algorithms for 0/1 Multidimensional Knapsack Problems.
6. Genetic Algorithm <http://en.wikipedia.org/wiki/Genetic_algorithm>
7. Pablo Moscato, On Evolution, Search, Optimization, Genetic Algorithm and Martial Arts towards Memetic Algorithms
8. Memetic Algorithms <http://en.wikipedia.org/wiki/Memetic_algorithm>
9. A Thesis by Fengjie Wu, A Framework for Memetic Algorithms
10. Pablo Moscato and Carlos Cotta, A Gentle Introduction to Memetic Algorithms