STUDY OF PROBABILSTIC APPROACH TO SOLVE 0/1 KNAPSACK PROBLEM

Ritika Mahajan*

Sarvesh Chopra**

Sonika Jindal***

ABSTRACT

The purpose of this paper is to study and analyze algorithm design paradigms applied to single problem – 0/1 Knapsack Problem. We consider situations in which a decision maker with a fixed budget faces a sequence of options, each with a cost and a value, and must select a subset of them so as to maximize the total value. The Knapsack Problem is a NP-complete problem and uses deterministic and probabilistic techniques to get solved.

Our objective is to analyze that how the probabilistic technique - Genetic Algorithm affect the performance of Knapsack Problem. Genetic Algorithm (GA) emulates biological evolution to solve a complex problem. GAs relies heavily on randomness. These algorithms basically search through a space of potential solutions using randomness as a major factor to make decisions. Instead of trying to solve the problem directly, they create random solutions and randomly mix them up until a good solution is found. Our experimental results show that the genetic algorithm is the promising approach as it gives result in optimal time.

Keywords: *Knapsack Problem, Probabilistic technique, Genetic Algorithm*

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

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

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

1. INTRODUCTION

The Knapsack Problem is a combinatorial optimization problem where one has to maximize the benefits of objects in a knapsack without exceeding its capacity. It is an NP-complete problem and uses deterministic and probabilistic techniques to get solved.

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 packings. [1]

We have *n* kinds of items, 1 through *n*. Each kind of item *i* has a value *vi* and a weight *wi*. All values and weights are nonnegative. The maximum weight that we can carry in the bag is *W*.

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]

We consider situations in which a decision maker with a fixed budget faces a sequence of options, each with a cost and a value, and must select a subset of them so as to maximize the total value.

Consider the following problem:

A hitch-hiker wants to fill up his knapsack, selecting among various objects. Each object has a particular weight and obtains a particular profit. The knapsack can be filled up to a given maximum weight. How can he choose objects to fill the knapsack maximizing the obtained profit?

A naive approach to solve the problem could be the following:

Select each time the object which has the highest profit and fits in the knapsack, until you cannot choose any more objects.

Unfortunately, this algorithm gives a good (or bad) solution which, in general, is not optimal. This is the case also if we try other rules, such as *"choose the object with the highest ratio profit/weight"*. [2]

2. GENETIC ALGORITHM

A genetic algorithm (GA) is a search technique used in computer science to find approximate solutions to optimization and search problems. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, natural selection, and recombination (or crossover). [3]

Genetic algorithms are typically implemented as a computer simulation in which a population of abstract representations (called chromosomes) of candidate solutions (called individuals) to an optimization problem evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but different encodings are also possible. The evolution starts from a population of completely random individuals and happens in generations. In each generation, the fitness of the whole population is evaluated, multiple individuals are stochastically selected form the current population (based on their fitness), modified (mutated or recombined) to form a new population, which becomes current in next iteration of the algorithm. [4]

GAs exploits the idea of the survival of the fittest and an interbreeding population to create a novel and innovative search strategy. A population of strings, representing solutions to a specified problem, is maintained by the GA. The GA then iteratively creates new populations from the old by ranking the strings and interbreeding the fittest to create new strings, which are (hopefully) closer to the optimum solution to the problem at hand. So in each generation, the GA creates a set of strings from the bits and pieces of the previous strings, occasionally adding random new data to keep the population from stagnating. The end result is a search strategy that is tailored for vast, complex, multimodal search spaces. [5]

Genetic algorithms are simple to implement, but their behaviour is difficult to understand. In particular it is difficult to understand why these algorithms frequently succeed at generating solutions of high fitness when applied to practical problems. The building block hypothesis (BBH) consists of: [6]

1. A description of a heuristic that performs adaptation by identifying and recombining "building blocks", i.e. low order, low defining-length schemata with above average fitness.
2. A hypothesis that a genetic algorithm performs adaptation by implicitly and efficiently implementing this heuristic.

Genetic Algorithms (GA) have gained a lot of popularity due to its ease of implementation, and the robustness of its search and have been applied successfully in a variety of fields, even

though special care has to be taken for highly structured combinatorial optimization problems.

There is no universally accepted definition of Genetic algorithms in the Evolutionary computation community. Some of the standard definitions are- [7]

Goldberg defines as: Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics.

Mitchell defines as: Genetic algorithms have at least the following elements in common: populations of chromosomes, selection according to fitness, crossover to produce offspring, and random mutation of new offspring.

Natural	**Genetic Algorithm**
Chromosome	String
Gene	Feature, character, or detector
Allele	Feature value
Locus	String position
Genotype	Structure
Phenotype	Parameter set, alternative solutions, a decoded structure
Epitasis	Non-linearity

Table1 Comparison of Natural and Genetic Algorithm Terminology

3. BASIC ELEMENTS OF GAS

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 requires 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. Moreover, to increase the performance of GAs, the selection methods are enhanced by elitism. Elitism is a method, which first copies a few of the top scored chromosomes to the new population and then continues generating the rest of the population. Thus, it prevents loosing few best found solutions.

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 sub sequences before and after that locus between two chromosomes to create two offspring. For example, consider the following parents and a crossover point at position 3:

Parent 1 1 0 0 | 0 1 1 1

Parent 2 1 1 1 | 1 0 0 0

Offspring 1 1 0 0 1 0 0 0

Offspring 2 1 1 1 0 1 1 1

In this example, Offspring 1 inherits bits in position 1, 2, and 3 from the left side of the crossover point from Parent 1 and the rest from the right side of the crossover point from Parent 2. Similarly, Offspring 2 inherits bits in position 1, 2, and 3 from the left side of Parent 2 and the rest from the right side of Parent 1.

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. Mutation can occur at each bit position in the string with some probability, usually very small (e.g. 0.001). For example, consider the following chromosome with mutation point at position 2:

Not mutated chromosome: 1 *0* 0 0 1 1 1

Mutated: 1 *1* 0 0 1 1 1

The 0 at position 2 flips to 1 after mutation.

4. KNAPSACK PROBLEM BY GENETIC ALGORITHM

Simple generational genetic algorithm procedure:

1. Choose the *initial* population of individuals
2. Evaluate the *fitness* of each individual in that population
3. Repeat on this *generation* until termination (time limit, sufficient fitness achieved, etc.):
   a) Select the best-fit individuals for *reproduction*
   b) *Breed* new individuals through *crossover* and *mutation* operations to give birth to *offspring*
   c) Evaluate the individual fitness of new individuals
   d) Replace least-fit population with new individuals

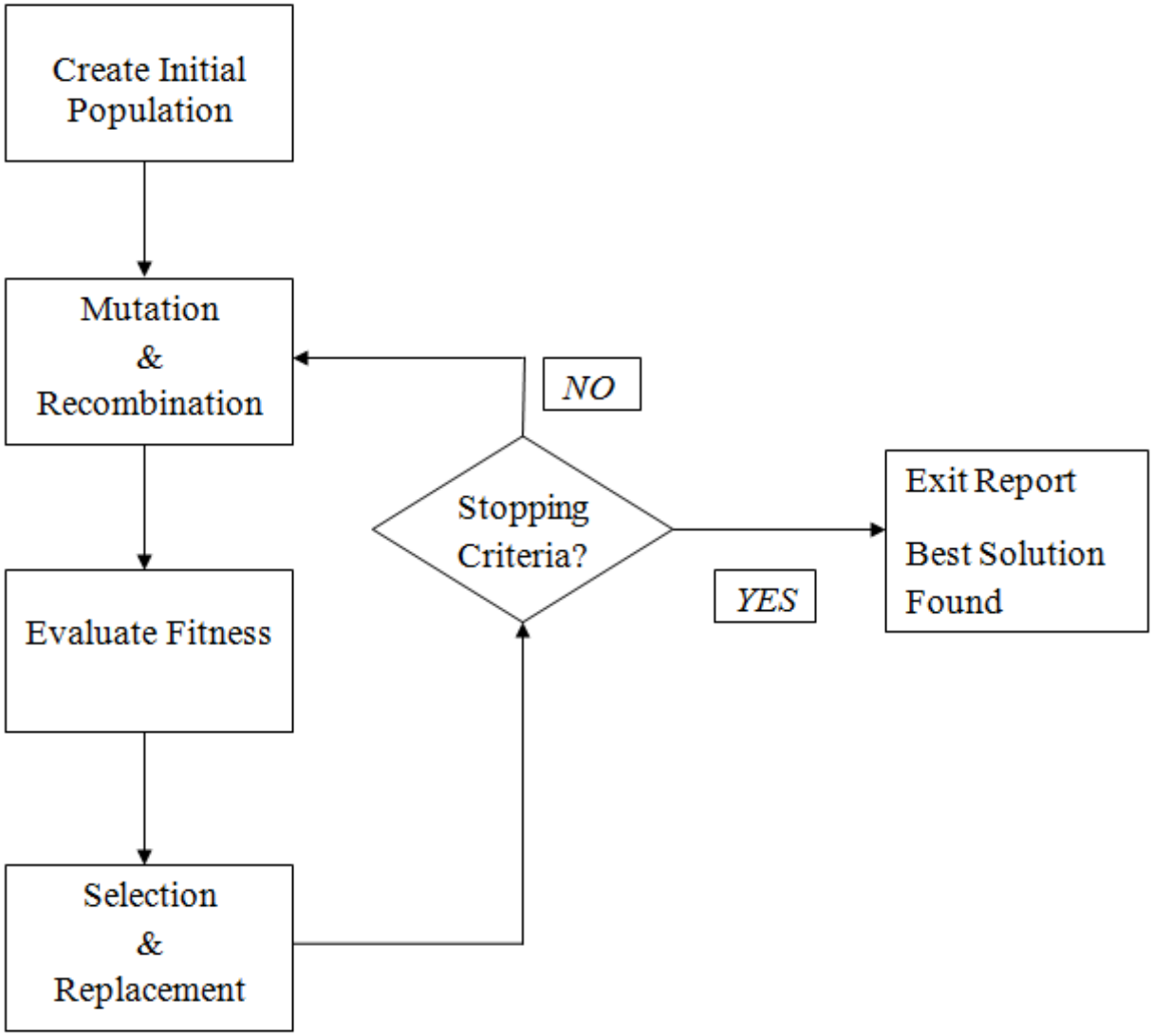

Fig. 1 Flowchart of Classical Genetic Algorithm

5. CONCLUSION

GAs reduces the complexity of the Knapsack Problem which makes it possible to find approximately optimal solutions for an NP problem.

Genetic algorithm is promising approach to solve knapsack problem as it gives better performance. It can be considered as best algorithm for finding the near to optimal solution for the most widely used NP-Complete problem i.e. Knapsack Problem if there is no constraint of memory.

6. ACKNOWLEDGMENT

We express our sincere gratitude to Mrs. Sonika Jindal who helped us immensely in completing this paper with her guidance.

7. REFERENCES

1. Knapsack problem. Online Available: http://www.utdallas.edu/~scniu/OPRE-6201/ documents/DP3-Knapsack.pdf
2. Knapsack problem. Online Available: http://en.wikipedia.org/wiki/Knapsack_ problem
3. David E. Goldberg, Addison Wesley, 1990. "*Genetic Algorithms in Search, Optimization and Machine Learning*"
4. Arild Hoff, Ingvar Mittet and Arne Løkketangen, Genetic Algorithms for 0/1 Multidimensional Knapsack Problems.
5. R.Sivaraj and Dr.T.Ravichandran, An Efficient Grouping Genetic Algorithm
6. Genetic algorithm. Online Available: http://en.wikipedia.org/wiki/Genetic_algorithm
7. Vincent Poirriez, Nicola Yanev, Rumen Andonov "*A Hybrid Algorithm for the Unbounded Knapsack Problem*" , Author manuscript, published in "Discrete Optimization (2008)" DOI : 10.1016/ j.disopt.2008.09.004
8. NP-complete. Online Available: http://en.wikipedia.org/wiki/NP-complete
9. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Riverst, Clifford stein "*Introduction to Algorithms*", second edition
10. Shailendra Kumar, June 2009 "Choosing Best Algorithm Design Strategies for a Particular Problem", Thapar university.