Genetic Algorithm based Approach to Solve Non Fractional (0/1) Knapsack Optimization Problem

Vikas Thada
Asst. Prof (CSE), ASET, Amity University,
Gurgaon, India

Shivali Dhaka
Asst. Prof (CSE), ASET, Amity University,
Gurgaon, India

ABSTRACT
In this paper we solve the non fractional knapsack problem also known as 0-1 knapsack using genetic algorithm. The usual approaches are greedy method and dynamic programming. Its an optimization problem where we try to maximize the values that can be put into a knapsack under the constraint of its weight. We solve the problem using genetic algorithm in matlab using gatool. In this research work different selection schemes have been used like roulette wheel, tournament selection, Stochastic selection etc. Following the introduction of genetic algorithm and knapsack problem, formulation of 0-1 knapsack problem in genetic algorithm is presented. Experimental results using various selection schemes have been analyzed and comparison of genetic algorithm technique is done with greedy method and dynamic programming optimizing techniques.

General Terms
Weight, value, optimization, problem

Keywords
Knapsack, genetic, algorithm,

1. INTRODUCTION
In this research paper solution of 0-1 knapsack problem is presented using Genetic Algorithm approach. The other approaches for solving are greedy method and dynamic programming. The greedy approach does not always results in an optimal solution and dynamic programming method having time complexity of O(nW) where n is number of items and W is the capacity of the knapsack. The reason for choosing this problem as this research work is that it is the problem that has been studied for almost a century and still finds its place in one of the good combinatorial optimization problem. In the problem number of items each with a weight and value are given. The aim is to find each item to be put in a knapsack so that the total weight of included items is less than or equal to the capacity of the knapsack simultaneous total value of the included items should be maximum. It's a problem that belongs to the NP class of problems. The decision problem form of the knapsack problem is NP-complete whereas optimization problem is NP-hard. As compare to other two methods mentioned above the GA based methods takes very less time and returns results in an efficient manner.

2. GENETIC ALGORITHM
GAs is search algorithms that follow the concept of natural selection and genetics [1]. GA are powerful and very efficient search and optimization techniques motivated by the natural selection theory of Darwin [2].

Genetic Algorithms **[3]** are based on the principle of heredity and evolution which claims "in each generation the stronger individual survives and the weaker dies". Therefore, each new generation would contain stronger (fitter) individuals in contrast to its ancestors. The process of GA's is iteration based of constant population size of candidate solutions. In each generation/iteration each chromosome's fitness in the current population is evaluated and new population evolves. Chromosomes with higher fitness values goes through reproduction phase in which selection, crossover and mutation operators are applied to get new population. Chromosomes with lower fitness values are discarded. Again this generated new population is evaluated and selection, crossover, mutation operators are applied. This process continues until we get an optimal solution for the given problem

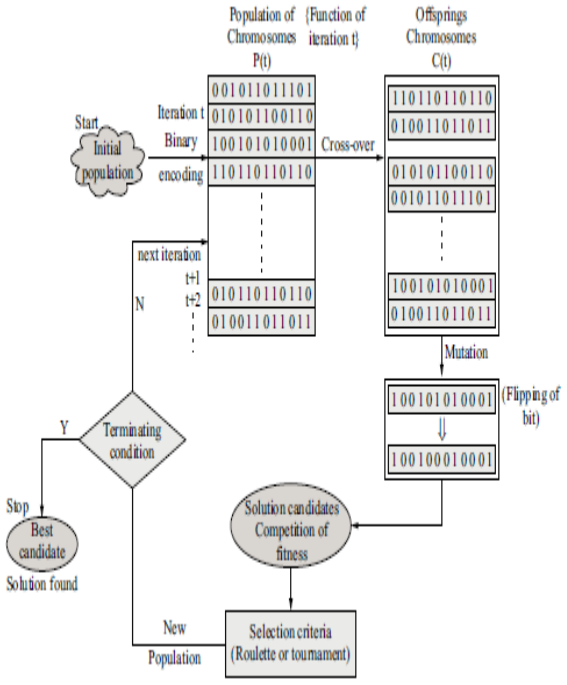

Fig 1: Basic operation of Genetic Algorithm[6]

2.1 Fitness Evaluation
Fitness function is a function which is responsible for evaluating some value to indicate among number of solutions which one is optimum. It can also be considered as a measure of performance or fitness to show how fit is the candidate solution. Fitness function for this research work will be discussed in the next section.

2.2 Selection

Once the fitness evaluation process is done next step is to perform selection operation. Process of selection operation is based on the principle of ‘‘survival of the fittest’. Higher fitness valued chromosomes goes through reproduction. Lower fitness valued chromosomes are discarded. There are number of ways to implement this operator, but all relies on the concept that candidates with good fitness values are to be preferred over poor fitness values. The idea is to give preference to better individuals. This selection operation does the replication of candidate chromosomes with good fitness values and eliminating those with poor fitness values [4].

Individual	Fitness
A	8
B	7
C	3
D	5

⇒

Individual	Fitness
A	8
B	7
A	8
D	5

Fig 2: Selection Operator on a Population of 4 Individuals[4]

The research work uses roulette wheel selection method, tournament selection method, stochastic uniform, uniform and remainder method as selection operator.

2.3 Crossover[1,5]

In the crossover operation mating of two chromosomes is performed that gives birth to two new offspring. This operation of crossover always happens with one parameter that is known as probability of crossover (ProC). When ProC is say 0.8 it means only 80% of the total population goes for crossover operation. Rests 20% chromosomes remain abstain from this operation and has no effect of crossover. Motive behind performing crossover operation is to explore new solutions and exploit use of old solutions. GA forms an optimum solution by mating two fit chromosomes together. Chromosomes with higher fitness will always have good selection probability then others with lower fitness values, thus a good solution moves from one generation to next generation

One point crossover (crossover point 7)

Doc1	**1**	**0**	**1**	**0**	**0**	**0**	**1**	0	0	0	0	1	0	0	0	0	0	1	0	0	0
Doc2	0	0	1	0	0	0	1	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

After crossover

Doc1	1	0	1	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	1
Doc2	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	

Fig 3: Single Point Crossover Explained

2.4 Mutation[1,5]

Mutation involves changing one bit of a chromosome from 0 to 1 or viceversa. This is performed under the constraint parameter called probability of mutation (ProM). For example if ProM is 0.10 then 10% genes of total chromosomes will go for mutation. The concept of mutation is based on this natural theory that varying breeds are possible only by varying gene values. After this operation fitness quality of new chromosomes may be high or low then old ones. In case new chromosomes are poor then old ones they are removed during selection process. The motive behind mutation is regaining the lost and discovering varying breeds. For example: randomly mutate chromosome at position 5

Doc1	1	0	1	0	**0**	0	1	0	1	0	0	1	0	0	0	0	0	1	0	0	0

After mutation (bit number 5 is mutated i.e. bit is negated)

Doc1	0	0	1	0	**1**	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1

Fig 4: Mutation Operation Explained

3. KNAPSACK PROBLEM

In the 0-1 knapsack problem given some items say n and a knapsack, the aim is to pack the knapsack to get the maximum total value. Each item has some weight and some value or profit. Total weight that we can carry is no more than some fixed number W that is the maximum weight knapsack can carry. So we must consider weights of items as well as their values. The aim is to fill the knapsack using various items so that the total weight of the items does not exceed the capacity of the knapsack i.e. W simultaneously maximizing the total profit of the included objects. The problem is called a 0-1 problem, because each item must be entirely accepted or rejected. Every object has a weight $\mathbf{w_i}$ and profit $\mathbf{p_i}$. The goal is to maximize the value/profit of the included objects in the knapsack. The value of $\mathbf{x_i}$ will be 0 if object is not included else $\mathbf{x_i}$ will be 1. Mathematically the problem can be stated as shown in equation 1.

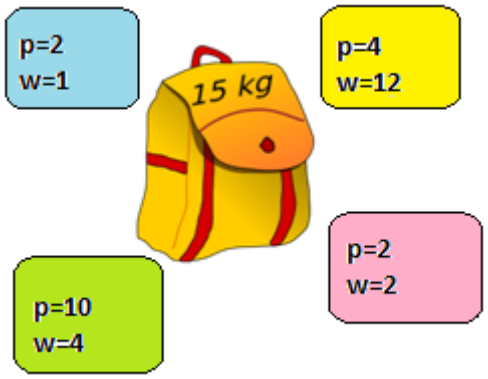

Fig 5: Knapsack problem with knapsack and four items

$$\boldsymbol{Maximize} \sum_{i=1}^{n} \boldsymbol{xi} * \boldsymbol{pi} \ \boldsymbol{subject\ to} \sum_{i=1}^{n} \boldsymbol{xi} * \boldsymbol{wi} \leq \boldsymbol{W} \quad \textbf{(1)}$$

Table 1: Different Solutions to 0-1Knapsack Instance shown in table 1.

x1	x2	x3	x4	x5	$\sum_{i=1}^{5} xi * pi$	$\sum_{i=1}^{5} xi * wi$
0	0	0	0	0	0	0
0	0	0	0	1	50	5
0	0	0	1	0	40	4
0	0	0	1	1	90	9
0	0	1	0	0	15	1
0	0	1	0	1	65	6
0	0	1	1	0	55	5
0	0	1	1	1	105	10
0	1	0	0	0	20	2
0	1	0	0	1	70	7
0	1	0	1	0	60	6
0	1	0	1	1	110	11
0	1	1	0	0	35	3
0	1	1	0	1	85	8
0	1	1	1	1	125	12
1	0	0	0	0	25	3
1	0	0	0	1	75	8
1	0	0	1	0	65	7
1	0	0	1	1	115	12
1	0	1	0	0	65	4
1	0	1	0	1	90	9
1	0	1	1	0	80	8
1	0	1	1	1	130	13
1	1	0	0	0	45	5
1	1	0	0	1	95	10
1	1	0	1	0	85	9
1	1	0	1	1	135	13
1	1	1	0	0	60	6
1	1	1	0	1	110	11
1	1	1	1	0	100	10
1	1	1	1	1	150	15

Here i denotes ith item, pi is profit of ith item, wi is weight of the ith item, xi is either 0 or 1 and W is capacity of knapsack.

3.7.1 Example of 0-1 Knapsack Problem

Let's consider the knapsack problem instance as given in table 1. The capacity W of knapsack in this instance is 6. Mathematically stating this instance of knapsack we have

$$Maximize \sum_{i=1}^{5} xi * pi = \\ = 25x1 + 20x2 + 15x3 + 40x4 + 50x5$$

Here each xi is either 0 or 1.

Subject to Constraint

$$Minimize \sum_{i=1}^{5} xi * wi = 3x1 + 2x2 + x3 + 4x2 + 5x5 \leq 6$$

Again each xi is either 0 or 1.

Table 2: An Instance of 0-1 Knapsack Problem

Item i	Profit pi	Weight wi
1	**25**	**3**
2	**20**	**2**
3	**15**	**1**
4	**40**	**4**
5	**50**	**5**

In the table 1 total possible solutions 2^5=32 are shown. The red marked solution violates the constraint in which total weight of included item exceeds capacity of knapsack. Optimal solution for this instance of 0-1 knapsack problem is 65 where item 3 and 5 are included in the knapsack.

4. MATLAB IMPLEMENTATION

For implementation of 0-1 knapsack problem in matlab the research work uses gatool built-in matlab. This is shown in figure:

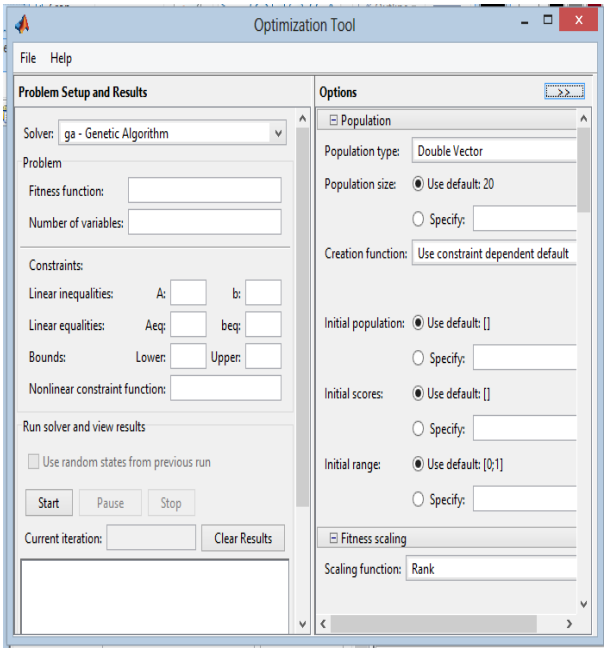

Fig 6: MATLAB GA optimization tool

The tool can be invoked in matlab using command: optimtool('ga')

For the implementation of this research problem 3 separate functions are to be written: First function is for maximizing profit, second is for constraint satisfaction and third function is for creating initial population. All three functions are listed here;

4.1 Fitness Function

The fitness function will try to maximize total profit as obtained by including the items in the knapsack following the constraint.

```
function y = simple_KS(x)

y = (25*x(1)+20*x(2)+15*x(3)+40*x(4)+50*x(5));

end
```

Here x is a vector that takes values from the creation function. All binary values (total 32) constitute the population of the problem to be solved by GA.

4.2 Constraint Function

The constraint function will try to minimize the total weights of the included objects so that it is less than equal to capacity of the knapsack. The function for the instance of the knapsack shown in table 1 is given as:

```
function [c, ceq] = simple_constraint_KS(x)

c = [3*x(1)+2*x(2)+1*x(3)+4*x(4)+5*x(5)-6];
ceq = [];

end
```

Similarly to the input to fitness function vector is passed as input to constraint function. Again values for x are taken from the initial population as supplied by creation function.

4.3 Creation Function

In order to write your own creation function for gatool you must write it as par the syntax as listed in the documentation of matlab. The function is given below:

```
function rtn = KPop1(genomeLength, fitnessFn, options )
x=[
   0 0 0 0 0; 0 0 0 0 1; 0 0 0 1 0; 0 0 0 1 1; 0 0 1 0 0;
   0 0 1 0 1;0 0 1 1 0; 0 0 1 1 1; 0 1 0 0 0; 0 1 0 0 1;
   0 1 0 1 0;0 1 0 1 1;0 1 1 0 0;0 1 1 0 1; 0 1 1 1 0;
   0 1 1 1 1;1 0 0 0 0; 1 0 0 0 1; 1 0 0 1 0;1 0 0 1 1;
   1 0 1 0 0;1 0 1 0 1;1 0 1 1 0;1 0 1 1 1;1 1 0 0 0;
   1 1 0 0 1;1 1 0 1 0;1 1 0 1 1;1 1 1 0 0;1 1 1 0 1;
   1 1 1 1 0;1 1 1 1 1;
   ];
rtn=x;
end
```

You simply need to pass the parameters as shown in the above manner and return the population as binary vector.

4.4 Termination Criteria

The terminating condition for the process is to either predefined number of generations reached or 97% of the population have same fitness value.

4.4 Results

The research work supplied the discussed functions to the gatool as shown below in figure X. The selection function was varied from one type to another and results were obtained. Different subsections of this section are solely because of different selection operators the research work used.

4.4.1 Stochastic Uniform Selection Operator

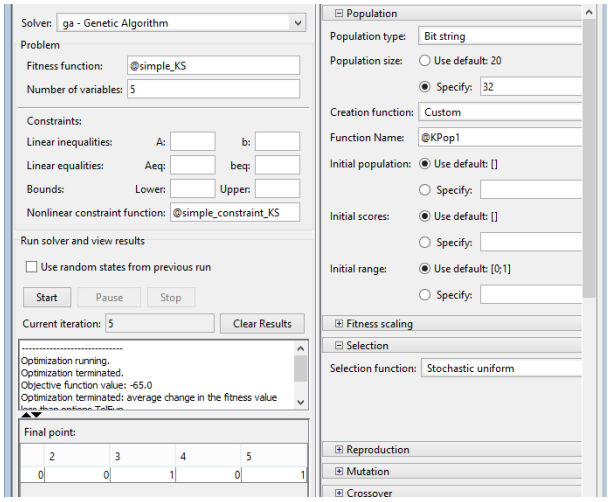

Fig 7: Sample Run and output for 0-1 knapsack problem

From the final point in the figure it is easily visible that for maximum profit item number 3rd and 5th should be selected so that maximum profit obtained in 65 and weight of the included object is 6 which satisfy the capacity constraint.

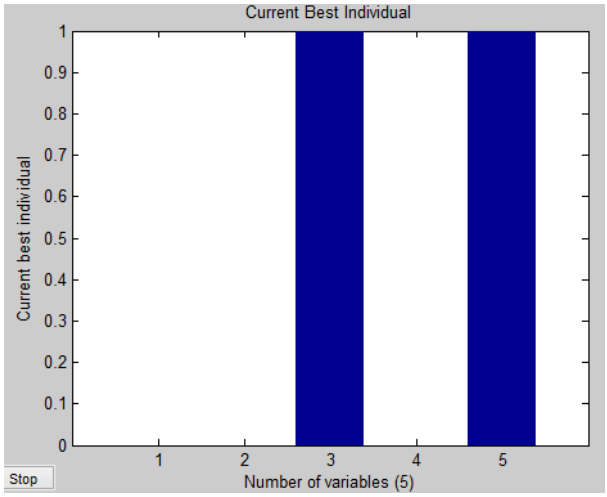

Fig 8: Current Best Individual using Stochastic Operator

From the figure it is also shown that current best individuals are item number 3 and 5 with value 1. This was also clear from display result on command window.

Generation	f(x)	constraint
1	-0	0
2	-50	0
3	-65	0
4	-65	0
5	-65	0

Figures for the remaining selection operators are same so we do not repeat the figures X1 and Y1.

4.4.2 Remainder Selection Operator

Results obtained as on command window:

Generation	f(x)	constraint
1	-0	0
2	-50	0
3	-65	0
4	-65	0
5	-65	0

4.4.3 Uniform Selection Operator

Results obtained as on command window:

Generation	f(x)	constraint
1	-0	0
2	-50	0
3	-65	0
4	-65	0
5	-65	0

4.4.5 Roulette Selection Operator

This selection operator didn't give us good results in the first run and process terminated after just 2 iterations but running the tool again for subsequent time gave us desired results as obtained using other selection operator.

Generation	f(x)	constraint
1	-0	0
2	-50	0
3	-65	0
4	-65	0
5	-65	0

4.4.6 Tournament Selection Operator

Results obtained as on command window:

Generation	f(x)	constraint
1	-0	0
2	-50	0
3	-65	0
4	-65	0
5	-65	0

5. CONCLUSION

The research work did not take into account croosover and mutation probability and used their default values from gatool. All the selection operators except roulette operator performed very well and optimal result was obtained in just 5 iterations. All the operators selected item 3 and 5 as item to be included in the knapsack. Thus it can be concluded that any selection operator except roulette wheel can be used for getting optimal results for 0-1 knapsack problem using gatool in matlab.

Comparing the technique with traditional dynamic programming approach it is concluded that even for more number of items knapsack problem can be easily solved without much complexity whereas dynamic programming method takes more amount of time and is less efficient.

6. REFERENCES

[1] B.Klabbankoh, O.Pinngern. "applied genetic algorithms in information retrieval" *Proceeding of IEEE* ,pp.702-711,Nov 2004

[2] S.S.Satya and P.Simon, "Review on Applicability of Genetic Algorithm to Web Search," *International Journal of Computer Theory and Engineering*, vol. 1, no. 4, pp. 450-455, 2009.

[3] Shokouhi, M.; Chubak, P.; Raeesy, Z *" Enhancing focused crawling with genetic algorithms"*Vol: 4-6, pp.503-508,2005.

[4] M.A.Kauser, M. Nasar, S.K.Singh, "A Detailed Study on Information Retrieval using Genetic Algorithm", *Journal of Industrial and Intelligent Information* vol. 1, no. 3, pp.122-127 Sep 2013.

[5] J.R. Koza, " Survey Of Genetic Algorithms And Genetic Programming*", Proceedings of the Wescon,* pp.589-595,1995

[6] V.Thada, V.Jaglan, "Use of Genetic Algorithm in Web Information Retrieval", *International Journal of Emerging Technologies in Computational and Applied Sciences*, vol.7,no.3,pp.278-281, Feb,2014