An Implementation of an Initial Scale in Solving Binary Knapsack Problem Using a Genetic Algorithm

Abbas Y. Al-Bayati **Nawar N. Qubat**
profabbasalbayati@yahoo.com
College of Computer sciences and Mathematics
University of Mosul/Iraq

Received on: 17/10/2006 **Accepted on: 24/12/2006**

ABSTRACT

In this paper, we used a new operation in a Genetic Algorithm for solving the binary Knapsack problem depending on it's LP Relaxation solution after eliminating the fractional part of the non-binary values. The benefit is to make a filter to the initial random population from the farness of the optimal solution and unsuitable chromosomes. This good property will be fixed automatically in all generations in the Genetic Algorithm until reaching the optimal binary solution.

Keywords: Genetic Algorithm, binary Knapsack problem, LP Relaxation solution.

استخدام عملية جديدة في الخوارزمية الجينية لحل مسألة الحقيبة الثنائية ذات القياس الابتدائي

عباس يونس البياتي **نوار نجم قوبات**
كلية علوم الحاسبات والرياضيات، جامعة الموصل

تاريخ استلام البحث: 2006/10/17 **تاريخ قبول البحث: 2006/12/24**

الملخص

في هذا البحث تم استخدام عملية جديدة في الخوارزمية الجينية لحل مسألة الحقيبة الثنائية بالاعتماد على الحل الخطي المتراخي لها بعد حذف الجزء ألكسري من القيم غير الثنائية. والفائدة هي لعمل تنقية للمجتمع العشوائي الأولي من الحلول البعيدة عن الحل الأمثل ومن الكروموسومات غير المفيدة. هذه الصفة الجيدة ستكون ثابتة تلقائيا في كل الأجيال في الخوارزمية الجينية إلى حين الوصول إلى الحل الثنائي الأمثل.

الكلمات المفتاحية: الخوارزمية الجينية، مسألة الحقيبة الثنائية، الحل الخطي المتراخي.

1. Introduction to Genetic Algorithm:

A Genetic Algorithm (GA) was first introducted by John Holland for the formal investigation of the mechanisims of natural adaptation but the algorithm has been since modified to solve computational search problems, GA has been used to solve the NP-hard combinatorial problems effectively such as Knapsack Problem.[2]

GA is a heuristic search algorithm for the solution of optimization problems in which starting from a random initial guess solution, better

descendants are tried in an attempt to find one that is the best under some criteria and conditions.[5]

GA is a computer algorithm that searches for good solutions to a problem from among a large number of possible solutions. It begins with a set of candidate solutions (chromosomes) called population. A new population is created from solutions of an old population in hope of getting a better population. Solutions which are chosen to the new solutions (offspring) are selected according to their fitness. The more suitable solutions are the bigger chances they have to produce.

The chromosomes in GAs represent the space of candidate solutions. possible chromosomes encoding are binary, permutation, value, and tree encodings [1]. For the Knapsack Problem we use binary encoding, where every chromosome is a string of bits "0" or "1".[4]

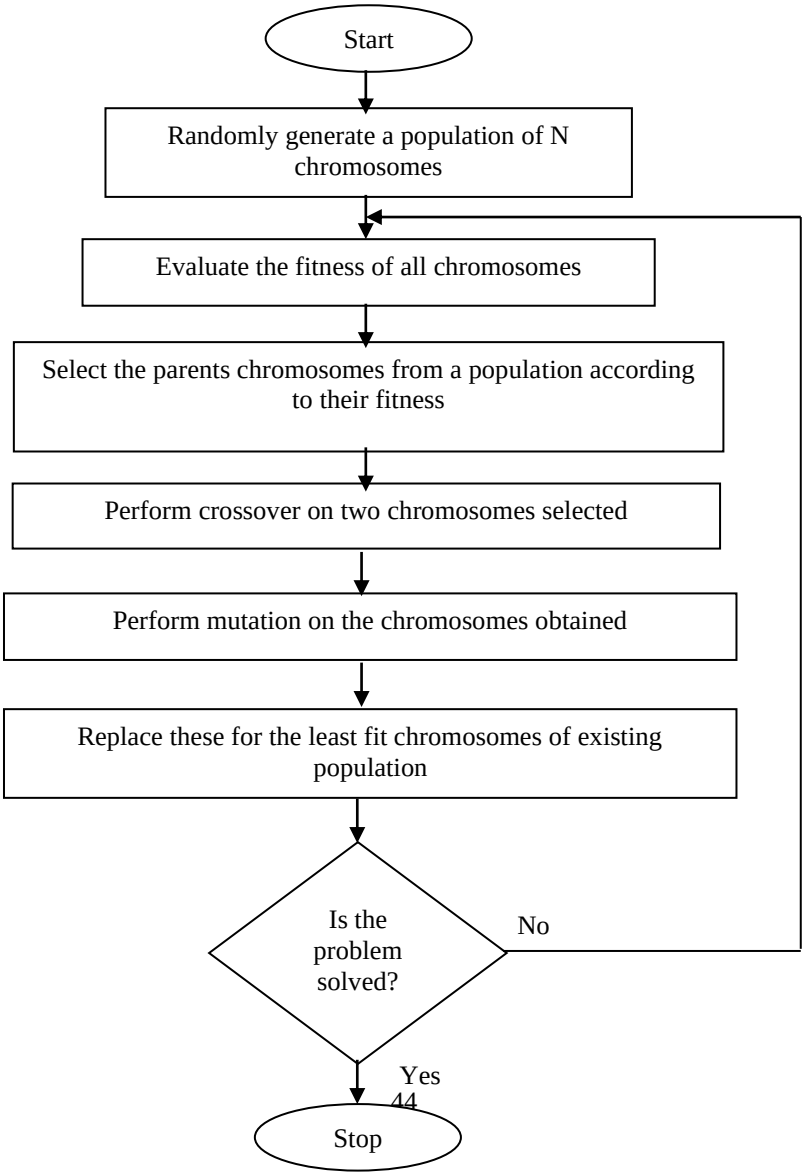

Figure (1): Flowchart of the General Genetic Algorithm [3]

2. Basic operations for Genetic Algorithm:

The basic operations for Genetic Algorithm are, see figure (1), and buttom some outlines of those operations.

Population: A population of possible solutions is randomly created. As stated before, this will be a population of random sets of 1's and 0's. Weighting more 1's or more 0's is problem-specific.

Fitness: In order to decide which solutions or chromosomes are better, the idea of fitness is introduced. Each possible solution has a fitness calculated by a fitness function. This fitness function must take into account what the actual problem is asking for and derive some sort of formula to decide how close each chromosome is to solving the problem. The form of the fitness function we used in this paper is $(f(x_j)/\sum_{j=1}^{n} f(x_j))$, where $f(x)$ is the objective function.

Selection: The selection operation will find very fit chromosomes to be used as parents of the next generation. Also, since not only the fittest of chromosomes should be able to reproduce, there must be some randomness involved in parents selection .Here are some kinds of selection operation.

(i) Roulette-wheel selection:

Parents are selected according to their fitness. The better chromosomes are the more chances to be selected they have. Imagine a roulette wheel where all chromosomes are placed in the population occupying space according to their fitness function. Then marble is thrown there and it selects the chromosome. A chromosome with bigger fitness will be selected for more times [1].

(ii) Linear rank selection:

In the beginning, the potentially good individuals sometimes fill the population too fast which can lead to premature convergence into local maxima. On the other hand, refinement in the end phase can be slow since the individuals have similar fitness values. These problems can be overcome

by taking the rank of the fitness values as the basis for selection instead of the values themselves.

(iii) Tournament selection:

In this scheme, a small group of individuals is sampled from the population and the individual with best fitness is chosen for reproduction. This selection scheme is also applicable when the fitness function is given in implicit form, i.e. when we only have a comparison relation which determines which of the two given individuals is better.[2]

Another idea often used is **Elitism**. This means that the best chromosomes of a generation automatically used to the next generation. This is especially useful to assure that, at worst, the next generation will have a solution as good as the last generation [7].

Crossover: When referring to reproduction in genetic algorithms, the word crossover is used. If you consider that reproduction is really a "crossing over" of two parent's genes into a child, this makes perfect sense. Crossover is preformed by taking two parents chromosomes solutions and swapping a certain number of their bits.see figure (2).

One-point crossover: One crossover is selected, binary string from beginning of chromosome to the crossover point is copied from one parent and the rest is copied from the second parent:

Parent Chromosome 1: 0 0 0 1 1 0 **1 1 0 0 1 0**
Parent Chromosome 2: 0 1 0 0 0 1 **1 1 0 1 1 1**

Child Chromosome 1: 0 0 0 1 1 0 1 1 0 1 1 1
Child Chromosome 2: 0 1 0 0 0 1 1 1 0 0 1 0

Two-point crossover: Two crossover point are selected, binary string from beginning of chromosome to the first crossover point is copied from one parent, the part from the first to the second point is copied from the second parent and the rest is copied from the first parent:

Parent Chromosome 1: 0 0 **0 1 1 0 1 1 0** 0 1 0
Parent Chromosome 2: 0 1 **0 0 0 1 1 1 0** 1 1 1

Child Chromosome 1: 0 0 0 0 0 1 1 1 0 0 1 0
Child Chromosome 2: 0 1 0 1 1 0 1 1 0 1 1 1

Uniform crossover: Bits are randomly copied from the first or from the second parent:

Parent Chromosome 1: 0 **0 0** 1 **1** 0 1 **1 0 0** 1 0

Parent Chromosome 2: 0 **1 0** 0 **0** 1 1 **1 0 1** 1 1

Child Chromosome 1: 0 1 0 1 0 0 1 1 0 1 1 0
Child Chromosome 2: 0 0 0 0 1 1 1 1 0 0 1 1

Arithmetic crossover: Some arithmetic operations are performed to make a new child:

Parent Chromosome 1: 0 0 0 1 1 0 1 1 0 0 1 0
Parent Chromosome 2: 0 1 0 0 0 1 1 1 0 1 1 1

Child Chromosome 1: 0 0 0 0 0 0 1 1 0 0 1 0 (**AND**)
Child Chromosome 2: 0 1 0 1 1 1 1 1 0 1 1 1 (**OR**)

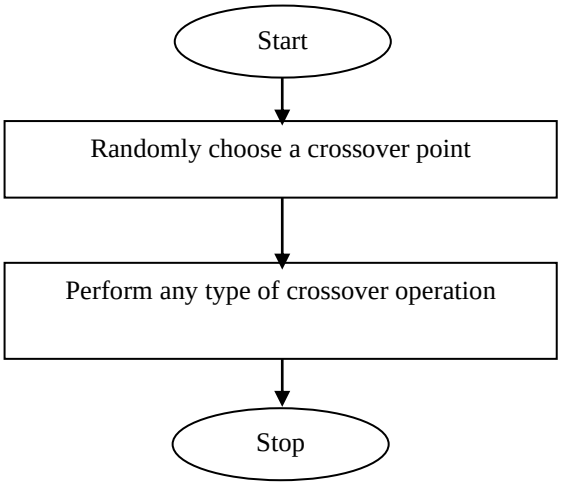

Figure (2): Flowchart of general crossover operation [3]

Mutation: Mutation is another genetic operation that introduces more randomness to a population. An example of mutation in nature would be albino animals. These animals have an altered genetic make-up that changes their fitness in their environment. GA mutation works the same way. Mutation changes the new child by flipping bits from 1 to 0 or 0 to 1.[7], see figure (3):

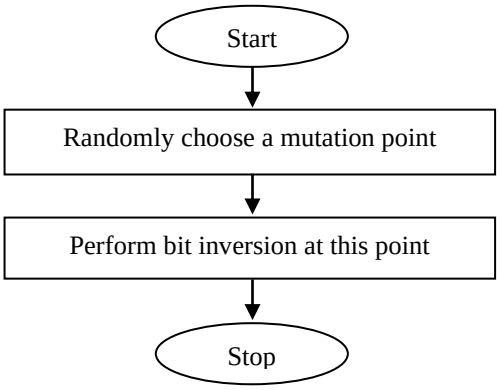

Figure (3): Flowchart of mutation operation [3]

3. Binary Knapsack problem:

The Binary Knapsack Problem is one of the most important model in combinatorial optimization, having numerous real-life applications as well as being an important subproblem in several solution algorithms for more complex problems.

Assume that n items are given, each item having a nonnegative integer profit p_j and weight w_j, for $j = 1,2,...,n$. The problem is to select a subset of the items, so that their overall profit is maximized without using the overall weight to exceed a given capacity C. Formally we may define the problem as:

Maximize $z = \sum_{j=1}^{n} p_j x_j$ …….……(1),

subject to $\sum_{j=1}^{n} w_j x_j \le C$ ……..……..…(2),

where $x_j \in \{0,1\}$, $j = 1,2,...,n$ ……(3)

$C \ge 0$, integer

where the binary variable $x_j = \begin{cases} 1 & \text{If item } j \text{ is selected} \\ 0 & \text{Otherwise} \end{cases}$

The optimal solution is denoted x^* and the corresponding solution value z^*.[6]

4. Linear Program Relaxation of Binary Knapsack Problem:

Solving the linear program of Knapsack is simply by filling the Knapsack with the most efficient items according to:

1-Decreasing p_j / w_j until the first item $s \in \{1,2,...,n\}$ does not fit into the Knapsack.

2-The weight w_j of the first item $m \in \{1,2,...,n\}$ does not fit into the Knapsack.

In this case a suitable fractional of items is chosen so that all the remaining capacity is used. The items will be denoted by the **split** item. The

solution becomes $x_j = 1$ for $j = 1,2,..., m-1$ and $x_s = (C - \sum_{j=1}^{m-1} w_j)/w_m$, the corresponding solution value will be:

$$z_{LP} = \sum_{j=1}^{m-1} p_j + (C - \sum_{j=1}^{m-1} w_j) * (p_s / w_s) \quad \ldots(4)$$

The solution value z_{LP} to the linear problem is an upper bound on the solution value to the original Knapsack Problem. This means that $z^* \le z_{LP}$ for the optimal solution value z^*. Any feasible solution is, on the other hand, denoted a lower bound z. Obviously, we have $z \le z^*$.[6]

5. A Binary Knapsack Problems:

Problem (1):

A climber is preparing for an expedition to Mount Optimization. His equipment consists of 4 items, where each item j, $j = 1,...,4$ has a profit p_j and a weight w_j (according to the table bottom). The climber knows that he will be able to carry items of total weight at most (14) kg. He would like to pack his knapsack in such a way that he gets the largest possible profit without exceeding the weight limit.

Items j	1	2	3	4
Profit p_j	16	22	12	8
Weight w_j	5	7	4	3

Solution:

The Binary Knapsack Problem form will be:

Maximize $z = 16x_1 + 22x_2 + 12x_3 + 8x_4$

Subject to $5x_1 + 7x_2 + 4x_3 + 3x_4 \le 14$

Where $x_j \in \{0,1\}$, $j = 1,2,3,4$

Simply the first item which does not fit into the knapsack according to p_j / w_j is 3 (i.e. $s = 3$) and the first item which does not fit into the knapsack according to their weight w_j is 3 (i.e. $m = 3$) therefore by eq. (4):

$$z_{LP} = \sum_{j=1}^{2} p_j + (C - \sum_{j=1}^{2} w_j) * (p_3 / w_3)$$

=16+22+(14-5-7)(12/4)=38+2*(12/4)=38+6=44 the optimal linear value.

Also, the variables values are $x_1 = 1, x_2 = 1, x_3 = (14-12)/4 = 0.5$ and $x_4 = 0$.

Problem (2):

A company is considering six investments. Investment 1, 2, 3, 4,5 and 6 will yield \$40000, \$80000, \$10000, \$10000, \$4000 and \$20000 respectively. Each investment requires a certain capital at present time: Investment 1, \$40000; investment 2, \$50000; investment 3, \$30000; ;investment 4, \$20000; investment 5, \$10000 and investment 6, \$40000. At present, \$100000 in total is available. Formulate an IP to maximize the total yield.

Solution:

For each investment, the choice is either invest or not. This leads us to define for $j = 1,2,3,4,5,6$:

$$x_j = \begin{cases} 1; \text{ If investment } j \text{ is made} \\ 0; \text{ Otherwise} \end{cases}$$

Therefore, the Binary Knapsack Problem form will be:

Max $z = 40000x_1 + 30000x_2 + 10000x_3 + 10000x_4 + 4000x_5 + 20000x_6$

Subject to

$40000x_1 + 50000x_2 + 30000x_3 + 20000x_4 + 10000x_5 + 40000x_6 \le 100000$

Where $x_j \in \{0,1\},\ j = 1,2,3,4,5,6$

Now, the first item which does not fit into the knapsack according to p_j / w_j is 6 (i.e. $s = 6$) and the first item which does not fit into the knapsack according to their weight w_j is 3 (i.e. $m = 3$) , by eq. (4):

$$z_{LP} = \sum_{j=1}^{2} p_j + (C - \sum_{j=1}^{2} w_j) * (p_6 / w_6)$$

=40000+30000+(100000-40000-50000)*(20000/40000)=70000+5000 =75000

Also, the variables values are $x_1 = 1, x_2 = 1, x_3 = x_4 = x_5 = 0$ and $x_6 = (100000 - 40000 - 50000)/40000 = 0.25$.

6. Some known methods for solving BKP:

(i) Branch and Bound method: Branch and Bound (BB) is a well-known approach and general optimization technique which is used especially for NP-hard problems. BB depends on the LP Relaxation solution of the

problem and works by branching or partioning it into many smaller subsets (nodes) such that:

$x_j \geq \lceil 1 \rceil$ and $x_j \leq \lfloor 0 \rfloor$ ……..(5)

and this branching operation continues on the feasible solution until reaching to the optimal solution.

(ii) Dynamic Programming: Dynamic Programming (DP) is a mathematical optimzation technique used for making a series of interrelated decisions. Usally, a multi-stage decision process is transformed into a series of single-stage decision process.

$$\left.\begin{array}{ll} P[j,c] = \max (P[j-1,c],\ p_j + [j-1, c-w_j]) & \\ \text{For } j = 1,..., n \text{ and } 0 \leq c \leq C & \\ \text{And } P[0,c] = 0 & \text{for } 0 \leq c \leq C \\ P[j,c] = -\infty & \text{for } c < 0 \end{array}\right\} \quad \text{……(6)}$$

DP starts with a small portion of the problem and finds the optimal solution for this smaller problem. It then qradually enlarges the problem, finding the current optimal solution from the previous one, until the original problem is solved in its entirety.

For more detials about BB & DP see [8] & [9], respectively.

7. Genetic Algorithm with LP Relaxation:

Here we are trying to put a new operation in Genetic Algorithm depending on the LP Relaxation solution after elimination the fractional part from the non-binary values and **fixing the value** of the variable which has the biggest profit in the objective function, this operation works only in the first of the Genetic Algorithm exactly before the fitness step and in the initial random population and automatically the result of this operation is fixed in the other created generations in future under condition that this solution exists in the initial population. We called this algorithm LPG Algorithm and the following is its outlines.

Outlines of the new suggested LPG Algorithm:

Step 1:[Initial Population]: Population of possible solutions is randomly created.

Step 2:[Filtering]: Fix the chromosomes in step1 which have the value of the variable with the largest profit and delete the others which do not have this value.

Step 3:[Fitness]: Evaluate the fitness (function value) of each chromosome in the population.

Step 4:[New population]: Create a new population by repeating following steps until the new population is complete.

(a):[Selection]: Select the parents chromosomes from a population according to their fitness (the better fitness=the bigger chance to be selected).

(b):[Crossover]: Crossover operation is to create a new children (offspring) from the parents, If no crossover was performed, children are exact copy of parents.

(c):[Mutation]: Mutation operator is performed by selection of random bits of the chromosome and replacing the value of the bit from "0" to "1" or vice versa.

(d):[Accepting]: Place new children in a new population.

Step 5:[Replace]: Use new generated population for further run of algorithm .

Step 6:[Problem Solved?]: If the end condition is satisfied (like the number of generations or the generated maximum solution is repeated), stop, and return the best solution in current population else go to step 3.

8. Numerical Results using LPG Algorithm:

In Problem (1): The LP Relaxation solution after cutting the fractional part from the non-binary values will be: (1, 1, 0, 0) which will exist in the initial population.

Now, Since x_2 has the largest profit ($p_2 = 22$) therefore we fix only chromosomes in the initial random population with this value (i.e. bit2 =1) and delete the others, this is called filtering.

The initial random population will be as follows with size=4:

No.	Chromosome	Solution	Fitness
1	1 1 0 0	38	0.27
2	0 1 0 1	30	0.21
3	1 0 1 1	36	0.26
4	0 1 1 0	34	0.24

By filtering the population from the chromosomes without (bit2 =1) we get a new population with size=3 which is less than the initial random population size and we continue under this size, such that:

Generation 1			
No.	**Chromosome**	**Solution**	**Fitness**

1	1 1 0 0	38	0.37
2	0 1 0 1	30	0.29
3	0 1 1 0	34	0.33
Maximum Solution is 38			

By doing crossover operation between the chromosomes in generation 1 we get a new chromosome which builds the generation 2, such that:

Chromosome 1: 1 1 0 **0** } 1 1 0 1 with solution = 46
Chromosome 2: 0 1 0 **1** } 0 1 0 0 with solution = 22

Chromosome 1: 1 1 **0 0** } 1 1 1 0 with solution = 50
Chromosome 3: 0 1 **1 0** } 0 1 0 0 with solution = 22

Chromosome 2: 0 1 0 **1** } 0 1 0 0 with solution = 22
Chromosome 3: 0 1 1 **0** } 0 1 1 1 with solution = 42

And by doing mutation operation on the chromosome (0 1 0 **0**) to the new one (0 1 0 **1**) we get the next generation such that:

Generation 2			
No.	**Chromosome**	**Solution**	**Fitness**
1	0 1 0 0	22	0.23
2	0 1 1 1	42	0.44
3	0 1 0 1	30	0.31
Maximum Solution is 42			

Note: The reason of not taking the chromosomes: (1 1 0 1) and (1 1 1 0) is because their solutions are infeasible.

Again, by doing crossover operation on generation 2 we get a new generation such that:

Chromosome 1: 0 1 0 **0** } 0 1 0 1 with solution = 30
Chromosome 2: 0 1 1 **1** } 0 1 1 0 with solution = 34

Chromosome 1: 0 1 0 **0** } 0 1 0 1 with solution = 30
Chromosome 3: 0 1 0 **1** } 0 1 1 0 with solution = 22

Chromosome 2: 0 1 **1 1** } 0 1 0 1 with solution = 30
Chromosome 3: 0 1 **0 1** } 0 1 1 1 with solution = 42

Generation 3

No.	Chromosome	Solution	Fitness
1	0 1 1 0	34	0.32
2	0 1 0 1	30	0.28
3	0 1 1 1	42	039
Maximum Solution is 42			

From the generations 2 and 3 we got the same maximum solution 42 and this is the optimal solution for our problem and this is the stop criteria.

In Problem (2): The LP Relaxation solution after cutting the fractional part from the non-binary values will be: (1, 1, 0, 0, 0, 0) which will exist in the initial population.

Since x_1 has the largest coefficient in the objective function therefore we fix only chromosomes in the initial random population with this value (i.e. bit1 =1) and delete the others.The initial random population will be as follows with size=5:

No.	Chromosome	Solution	Fitness
1	1 0 0 0 0 0 1	60000	0.20
2	0 1 0 1 1 0	44000	0.15
3	1 0 0 0 1 1	64000	0.21
4	1 1 0 0 0 0	70000	0.23
5	0 1 0 0 1 1	54000	0.18

The new population after the filtering will become:

Generation 1			
No.	**Chromosome**	**Solution**	**Fitness**
1	1 0 0 0 0 0 1	60000	0.30
2	1 0 0 0 1 1	64000	0.32
3	1 1 0 0 0 0	70000	0.36
Maximum Solution is 70000			

Doing crossover operation between the chromosomes in generation 1 we get a new generation 2, such that:

Chromosome 1: 1 0 0 0 **0 1** } 1 0 0 0 1 1 with solution = 64000
Chromosome 2: 1 0 0 0 **1 1** } 1 0 0 0 0 1 with solution = 60000

Chromosome 1: 1 0 0 0 0 **1** } 1 0 0 0 0 0 with solution = 40000
Chromosome 3: 1 1 0 0 0 **0** } 1 1 0 0 0 1 with solution = 90000

Chromosome 2: 1 0 0 0 **1 1** } 1 0 0 0 0 0 with solution = 40000

Chromosome 3: 1 1 0 0 **0 0** 1 1 0 0 1 1 with solution = 74000

By doing mutation on the chromosome (1 1 0 0 1 **1**) to the new one (1 1 0 0 1 0) we get the next generation such that:

Generation 2			
No.	**Chromosome**	**Solution**	**Fitness**
1	1 0 0 0 1 1	64000	0.32
2	1 0 0 0 0 1	60000	0.30
3	1 1 0 0 1 0	74000	0.37
Maximum Solution is 74000			

Again, by doing crossover operation on generation 2 we get a new generation such that:

Chromosome 1: 1 0 0 0 **1 1** } 1 0 0 0 0 1 with solution = 60000
Chromosome 2: 1 0 0 0 **0 1** } 1 0 0 0 1 1 with solution = 64000

Chromosome 1: 1 0 0 0 1 **1** } 1 0 0 0 1 0 with solution = 44000
Chromosome 3: 1 1 0 0 1 **0** } 1 1 0 0 1 1 with solution = 94000

Chromosome 2: 1 0 0 0 **0 1** } 1 0 0 0 1 0 with solution = 44000
Chromosome 3: 1 1 0 0 **1 0** } 1 1 0 0 0 1 with solution = 90000

Generation 3			
No.	**Chromosome**	**Solution**	**Fitness**
1	1 0 0 0 0 1	60000	0.35
2	1 0 0 0 1 1	64000	0.38
3	1 0 0 0 1 0	44000	0.26
Maximum Solution is 64000			

We observe that the solution of generation 3 is worse than the solution of generation 2, therefore, we stop the process, and hence the optimal solution is 74000.

9. Numerical Discussion:

The new genetic algorithm using filtering property in Problem 1 and Problem 2 makes finite reduction for the initial population size such that for problem 1 the initial population size is reduced from 4 chromosomes to 3

and for problem 2 the initial population size is reduced form 5 chromosomes to 3, and this makes reducing the number of generations at least or equal the number of the generations under the general genetic algorithm.

10. Conclusion:

We have shown that how the genetic algorithm under filtering property can be used to find a good solution for the Binary Knapsack Problem. By using filtering the initial randomly population from the farness and unsuitable chromosomes by choosing only possible chromosomes under the LP Relaxation solution of the problem and this property will be fixed automatically in all generations until reaching the optimal binary solution.

This property makes the Genetic Algorithms more powerful for solving the binary problems and even for problems with a big size of generations. This will be done by reducing this size and by remaining the chromosomes with largest profit which guarantee the existness of the optimal solution by the benefit of the LP Relaxation of these problems.

REFERENCES

[1] Achrekar, H.; Gandhi, J. and Lakeshri, S. (2002) "Genetic Algorithms" , Student members, IEEE-VESIT, India.

[2] Bodenhofer, U., (2002) **Genetic Algorithms: Theory and Applications**, 2nd Edition, Johannes Kepler University, A-4040 Linz, Austria.

[3] Goldberg, D. E. (1989) **Genetic Algorithms in Search, Optimization and Machine Learning, The 0/1 Knapsack Problem**, Addison-Wesley Pub. Co., University of Illinois at Urbana-Champaign, ISBN: 0201157675.

[4] Hristakeva, M. and Shrestha, D.(2003) "Solving the 0-1 Knapsack Problem with genetic Algorithms", MICS, Proceedings, Techniqual reports, Simpson College, Indianola, Iowa No.(4).

[5] Nieminen, K.; Ruuth, S. and Maros, I. (2003) "Genetic algorithm for finding a good first integer solution for MILP", Systems Analysis Laboratory Helsinki University of Technology Istv an Maros Department of Computing Imperial College, London Department of Computing, Imperial College ISSN 1469—4174.

[6] Pisinger, D. (2003) "A toolbox for solving knapsack problem", DIKU, European Journal of Operational Research, University of Copenhagen, Denmark No.(88), Vol. (10), pp 122–145.

[7] Sastry, K. and Goldberg, D. and Kendall, G. (2005) **Genetic Algorithms**, chapter 4, in Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, Springer, pp 97-125.

[8] Terlaky, T. (2004) "Discrete Optimization..Branch and Bound Methods", Dept. of Computing and Software, McMaster University, Hamilton, Canada, Vol. (21), No.(6), pp. 207-236.

[9] Terlaky, T. (2004) "Discrete Optimization.. Dynamic Programming", Dept. of Computing and Software, McMaster University, Hamilton, Canada, Vol. (21), No.(7), pp. 175-180.