VOL 4 (2020) NO 1

e-ISSN : 2549-9904
ISSN : 2549-9610

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

Contrasting of Various Algorithmic Techniques to Solve Knapsack 0-1 Problem

Yogesh Awasthi[#], Ashish Sharma[*]

[#] *Department of Information Technology, Lebanese French University, College of Engineering and Computer Science, Erbil-KR, Iraq*
[*] *Department of Computer Networking, Lebanese French University, College of Engineering and Computer Science, Erbil-KR, Iraq*
E-mail: dryogeshawasthi@lfu.edu.krd, ashish.sharma@lfu.edu.krd

***Abstract*— This paper will point of convergence on a relative assessment and estimation of the dynamic programming, B&B, Greedy and Genetic algorithm including of the intricacy of time prerequisites, and the necessary programming endeavors and inspect the absolute incentive for every one of them. Out of these four, two algorithm (Greedy and Genetic) algorithm can be utilized to clear up the 0-1 Knapsack issue inside a sensible time multifaceted nature. The most pessimistic scenario time unpredictability (Big-O) of the two calculations is O(N). Parallelly, these calculations can't find the accurate response to the issue; they are valuable in detecting a close by premier final product as it were. Our basic commitment directly here is to investigate the two calculations contrary to common benchmark realities units and to quantify the precision of the impacts provided by method for each calculation. In this way, we will think about the top-notch neighbourhood result created by utilizing the calculation against the genuine real most dependable outcome.**

***Keywords*— Running Time, Complexity, B&B, Genetic, Greedy, DP.**

I. Introduction

The knapsack is an issue in combinatorial streamlining: given a lot of things, each with a weight and a worth, decide the quantity of every thing to remember for an assortment so the all-out weight is not exactly or equivalent to a given farthest point and the complete worth is as huge as could be expected under the circumstances. It gets its recognition from the issue looked by somebody who is compelled by a fixed-size rucksack and must select the bag with the most substantial items. The most well-known backpack issue is the paired (0–1) rucksack issue, where the leader is permitted to pick (1) or not to pick (0) the thing, at the end of the day, the things are not dividable. The 0/1 Knapsack Problem is a case of a combinative enhancement issue, which appears for a exceptional arrangement from amongst numerous extraordinary arrangements. It is worried about a knapsack that has wonderful entire number volume (or limit) V. There are n precise matters that may also conceivably be put in the backpack. Thing I has a fine total range quantity Vi and nice total quantity advantage Bi. In expansion, there are Qi duplicates of thing I accessible, the place quantity Qi is a two high-quality variety pleasing 1 <= Qi <= Infinity. Let Xi decides what range of duplicates of component I are to be set into the rucksack the objective is to:
Maximize

$$\sum_{i=1}^{N} RiXi \qquad (1)$$

Subject to the constraints

$$\sum_{i=1}^{N} ViXi \leq V \qquad (2)$$

and

$$0 \leq Xi \leq Qi \qquad (3)$$

In the event that at least one of the Qi is unending, the KP is unbounded; something else, the KP is limited [1].The limited KP can be either 0-1 KP or multi requirement KP. In the event that Qi = 1 for I = 1, 2, …, N, the issue is a 0-1 backpack issue In the present paper, we have chipped away at the limited 0-1 KP, where we can't have more than one duplicate of a thing in the knapsack(Gossett & Eric 2003).

II. The Knapsack Problem (KP)

The KP issue can be broadly applied in flotsam and jetsam classification, valuable asset portion, work planning, capital planning, venture choices, task choice, freight pressing and various fields. For this issue, its answer strategies can be separated into two classes: exactness calculations, (for example, thorough pursuit, dynamic programming strategy, branch and sure technique, and so on.) and estimate calculations, (for example, voracious strategy, hereditary calculation, subterranean insect calculation, and so on.) [5]. Since the KP issue has a place with the NP-C (Non-deterministic Polynomial Completeness) issue [6], its computational multifaceted nature is O(2n). In this paper, a 0/1 KP is as an answer of the 0/1 backpack issue, getting a handle on calculation, dynamics programming calculation, B&B calculation, and Genetic calculation are employed and assessed each systematically and tentatively as far as time and the total expense for every one of them, Moreover, a near investigation of the getting a handle on all four discussed algorithm and its calculations is displayed.

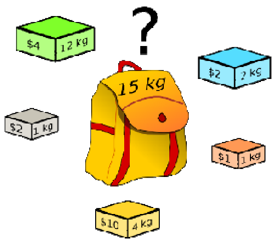


Fig. 1 0/1 Knapsack Problem

This paper is composed as pursues: Second part, gives a global perspective on foundation of knapsack issue, additionally exhibits the past connected work of the 0-1 KP and the calculations they are utilized to fathom it. Third segment of the paper contains the past work in this area. All calculations showed in fourth part. While in fifth part, expository perspective on calculation results will be displayed. Besides, the investigation includes the estimation of a few execution measurements, including: the most pessimistic scenario time intricacy. In sixth section, an examination of the exploratory outcomes between the four calculations will be appeared. At long last, the ends will be talked about in seventh segment.

III. Literature Review

Numerous Investigators has marked enforcing GA calculations to take care of 0/1 KP issues. Julstrom et. al. (2015) speak to the greedy calculations, genetic calculations and greedy genetic calculations penetrated the quadratic 0/1 rucksack issue. Here rucksack issue we need to detect customary backpack issue and characterizing the object of each and single article. Those outcomes show the force of hereditary calculations acquire on heuristic rule way to deal with gain ideal outcome on mix issue and to illuminate 0-1 rucksack issue utilizing genetic calculations.

G. Megha (2013) actualized an amended 0/1 backpack issue utilizing the combination of genetic and Hybrid Algorithm. Hereditary calculation is a computational calculation and quick, effective calculations to implement the 0-1 rucksack issue.

Umbarkar A.J. and Joshi M. (2014) present a cutting-edge way to deal with take care of 0-1 backpack issue utilizing Dual Population Genetic Algorithms. Double populace hereditary calculations are additionally giving ideal answer for the problem. The results speak to double populace hereditary calculations to improve and great execution in the 0-1 backpack issue, and check progressively troublesome rucksack issue.

Hristakeva M. and Shresthna D. proposed the usage of the 0/1 rucksack issue utilizing the Algorithm for genetic. We need to locate the ideal arrangement of the rucksack issue, and usage of these capacity roulette-wheel capacity and choice capacity for taking care of the issue.

Khuri et.al. (2012) speak to the usage of the 0-1 numerous rucksack issue utilizing hereditary calculations. Hereditary Algorithms utilizing for discipline furthermore, include of incomprehensible contribution to the populace for the hereditary calculations. The knapsack is an issue in combinatorial streamlining: given a lot of things, each with a weight and a worth.

IV. Different Approaches

A. *Greedy Algorithm*

Using this technology in optimization problems to make a decision is probably the right solution. We have three possibilities in this technique to figure out the 0/1 Knapsack problem

1) take the items that contain the highest value of others, this leads to increase the value of Knapsack immediately.
2) Take the lightest element in Knapsack, where many items are deleted.
3) Selection of high-weight items.

B. *Dynamic Programming*

Is a technique to solve sub - problems, where solve each small sub - problem once and stored only in memory so that the next time when we need the same solution can be easily found.

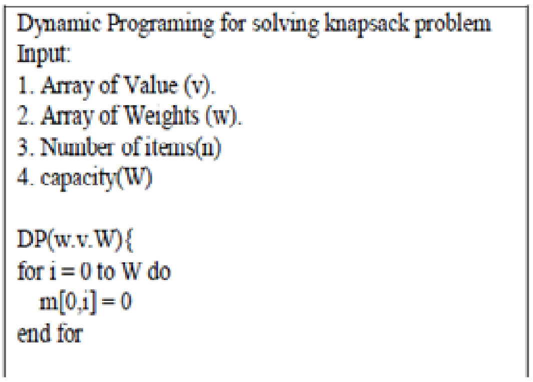
On the other hand, to find a solution to the problem, all of its sub-problems are solved separately. This sub-section is then assembled to obtain an ideal solution.

Let's the value of W [1 … N] and V [1 … N], structure of 2D-Array [0 ... N, 0 … Capacity]) of Dynamic Programming.

Subsequently, O (N*Capacity) shows the multifaceted nature of the Dynamic Programming calculation. When we define the DP as a memory requirement it requires 2D array which contain the rows as number of item columns as capacity of KP. This algorithm is likely one of the most comfortable to carry out because it does not demand the use of any extra anatomical structure.

C. *Branch & Bound Algorithm*

It is a direct technique for solving difficult problems in a holistic way. If it does not find values for the remaining



```
Dynamic Programing for solving knapsack problem
Input:
1. Array of Value (v).
2. Array of Weights (w).
3. Number of items(n)
4. capacity(W)

DP(w.v.W){
for i = 0 to W do
  m[0,i] = 0
end for
for i = 1 to n do
    for j = 0 to W do
        if w[i] ≤ j then
            m[i,j] = max (m[i-1,j],m[i-1,j-w[i]] + v[i])
        else
            m[i,j] = m[i-1,j]
        end if
    end for
end for
}
Return Max Value
```

```
Branch & Bound Pseudo code
Input:
Array of Weights and array of values
Output:
Max Value
Note: Items are sorted according to value/weight ratios
Queue Q
Node Type: current, temporary
*Create the root
Q.enqueue(root)
Max Value = value
While (Q is not empty)
    current = PQ.GetMax()
 if (current >MaxValue)
     Then Set the left child of the current node to include
the next item8
If child.Left value is greater than MaxValue
MaxValue = Value of the Left Child
End if
If child.left bound better than MaxValue
Q.enqueue(Left Child)
End if
If child.Right bound better than MaxValue
Q.enqueue(Right Child)
End if
Return Best solution
```

```
Greedy algorithm for solving knapsack problem (Weight []
and Benefit [])
Input:
1. Array for Weight, which holds the weight of all items.
2. Array for Benefit, which holds the Benefit of all items.
3. Capacity.
Output: Array Solution, which holds the items that its
weight does not surpass the knapsack and it had the
maximum amount of value
 A.      Calculate the ratio[1...n]
    1. n is the number of input items;
    2. W is the knapsack capacity;
    3. Weight [i] holds the weight of i^th item;
    4. Benefit [i] holds the Benefit of the i^th item;
    5. Calculate the Ratio value for each item :
    6.       for all input items do
    7.       Ratio[i] = Benefit[i]/weight[i];
 B.  Sort the items in a non-decreasing order according
     the items ratio value

    8.       $for(i = n-2; i >= 0; i--)$
    9.       $for(j = 0; j <= i; j++)$
    10.      $if(Benefit[j] < Benefit[j+1])$
    11.              Swap (Benefit [j+1] , Benefit [j]) ;
    12.              Swap (Weight[j+1] , Weight [j]) ;
    13.              Swap (Ratio[j+1] , Ratio [j]) ;
 C.      Knapsack algorithm (find the Knapsack solution)
  14. $for\ i = 1; i < n; i++\ )$
  15. $if\ (Weight[i] < W)\ then$
  16. $amount = Weight[i]$ ;
  17. $max\_value = max\_value + Benefit[i]$;
  18. $Solution[i] = amount$;
  19. else
  20. $amount = knapsack$;
  21. $solution[j++] = amount$ ;
  22. break;
  23. $W = W - amount$ ;
```

branches, which can give a solution, it will automatically ignore them, and solve the branches that have values, even if only one branch evaluates the solutions every time.

They use Branch and Bound calculation for the KP by showed which can secure either ideal or vague arrangements.

Best First Branch and Bound (Weights [1 … N], Values [1 … N).In the most pessimistic scenario, the branch and bound calculation will produce all single level stage and all leaves. In this manner, the tree would be generated and it has 2n-1 hubs, Lets say it will have an exponential intricacy. Notwithstanding, it is still superior to the animal power calculation on the grounds that all things considered it won't create every conceivable hub (arrangements). The necessary memory relies upon the length of the need line.

D. Genetic Algorithm

It is also called a computer algorithm, looking for the best solution among as many solutions as possible. Basic steps of algorithm are as.

Complexity: The multifaceted nature of the hereditary calculation relies upon the quantity of things (N) and the quantity of chromosomes in every age (Size). It is O(Size*N)

```
initialize the population
                current_generation := 1
                for i := 1 to N do
                          for j := 1 to L do
                                    p[i][j] := random_number(0, 1)
evaluate the population
                for i := 1 to N do
                          evaluate ( p[i] )
selection
                for i := 1 to N do
                          parents[i] := tournament ( p, tournament_size )
crossover
                for i := 1 to N step 2 do
                          for j := 1 to L do
                                    if ( j <= crossover_point )
                                              offspring[i][j] := parents[i][j]
                                              offspring[i+1][j] := parents[i+1][j]
                                    else
                                              offspring[i][j] := parents[i+1][j]
                                              offspring[i+1][j] := parents[i][j]
mutation
                for i := 1 to N do
                          if ( random_number < mutation_rate )
                                    mutation(offspring[i])
evaluate the offspring
                for i := 1 to N do
                          evaluate(offspring[i])
new population
                p := offspring
                current_generation += 1
termination condition
                if ( current_generation < max_generation )
                          return to step 3
the solution is the individual with the best evaluation
```

When comparing these three possibilities we find that the best results in the third possibility (selection of high-weight items)

V. Experimental Modelling

This section of the study shows the analytical or experimental modelling. For figuring out the 0/1 knapsack problem and finding the all effects generated by various situations the most important metric the show the performance of all four algorithms (DP, B&B, Greedy, Genetic algorithms). This study includes the running time parameter and performance to get maximum benefit to the knapsack. As we know the target of knapsack problem is to get the maximum profit.

The running time metric used to see that how much time is required and how much time is needed to finish the task assigned by the algorithm. On the other hand Time complexity evaluate the maximum time needed to solve the 0/1 rucksack problem over the unlike data items. The running time arrogates a immense component in increasing the function operation. By this fashion, the aim of any algorithm to solve 0/1 knapsack is to execute fertile effective result in the lowest existing time.

A. *Greedy Algorithm*

The running time complexity of greedy algorithm follows two steps as

1. First Sort by Merge sort algorithm is O(NlogN)
2. $\sum_{i=0}^{n} 1 \ to \ n-1 \ is \ O(N)$

Therefore the complexity of above algorithm is O (NlogN) + O(N) O(NlogN).

B. *Dynamic Algorithm*

Maximum running time taken by dynamic algorithm to find the solution of 0/1 rucksack problem is O(W×N).

C. *Branch & Bound Algorithm*

When B&B algorithm generates its all levels and nodes (in worst case) then the time complexity of the complete tree will be O(2N).

D. *Genetic Algorithm*

The array chromosomes has been introduced by the function of O(N). Fitness, Mutation and Crossover functions also have O (N). These two functions for selection have order 1. The termination condition checked by the function has order 1 and order of N is the total complexity of the program.

TABLE I
Time Complexicity Of Four Algorithm

Metric	Greedy	Dynamic	B&B	Genetic
Ex. Time	O(NlogN)	O(W*N)	$O(2^n)$	O(N)

TABLE II
SELECTION METHOD

items	0	1	2	3	4	5	6	7	8	9	10	11
Chr fitness	40	20	5	1	9	7	38	27	16	19	11	3
indexes	0	1	2	3	4	5	6	7	8	9	10	11

TABLE III
FITNESS ARRAY

items	0	1	2	3	4	5	6	7	8	9	10	11
indexes	0	6	7	1	9	8	10	4	5	2	11	3

TABLE IV
GENERATION METHOD

Population Size	Group Selection Method		
	No of Gen	Max fit found	Items chosen
100	39	3825	1,2,3,4,5,7,9,12
200	51	4310	1,2,3,4,5,6,7,8,11
300	53	4315	1,2,3,4,5,6,7,8,10
400	49	4320	1,2,3,4,5,67,8,9
500	65	4320	1,2,3,4,5,67,8,9
750	45	4320	1,2,3,4,5,67,8,9
1000	53	4320	1,2,3,4,5,67,8,9

VI. Experimental Result

All algorithms shown in the previous sections have been programmed in VC++. We run the all the programs by using variable array size but with constant capacity on processor AMD CORE i5 2.00 GHz and four gigabyte memory laptop and, we iterate the all algorithms forty times. After this process we get the average running time of the algorithms. For reading the data set we create a file generate the values between one to one thousand with variable sizes., yet we start the process to run the programs with least size array to check the code and correctness of the results. After that we make it for bigger one and find the results. Table 5 shows the running time calculated by the experimental programme for genetic, B&B, DP and Greedy algorithms with variable sizes (in thousands). We test the data up to the size of 60K due to the limitation in B&B algorithm because its complexity is O(2n) and it needs more space. The running time calculated the by the programme for all four algorithms has been shown in table 6.

The outcome of the programme as shown in table 6 are anticipated on the experimental model. The minimum time for out of all four algorithms belongs to genetic algorithm under the designed parameters and environment.

If we see other part of the algorithms we found that most likely results always measured by dynamic programming techniques yet other two i.e. greedy and genetic always evaluate the best local optimum result. Due to this reason we have implemented all the algorithms on the similar data block and check where they will obtain the most beneficial optimum outcome in course of running time.

When we focus on the table 7 and table 8 we analyze that local optimum result calculated by the genetic algorithm is better than the greedy one in most of the cases yet genetic local outcome is best as compare to the greedy outcome.

For evaluating the efficiency of all algorithms, the most important metric has been presented in this section.

TABLE V
Observational Time

Size	Greedy	Dynamic	B&B	Genetic
100K	0.8723	7.6177	NA	0.6325
200K	1.8758	13.8441	NA	1.769
300K	2.9489	21.1783	NA	2.2652
400K	3.9071	28.0098	NA	2.7336
500K	4.0682	34.4011	NA	3.5795

TABLE VI
OBSERVATIONAL TIME

Size	Greedy	Dynamic	B&B	Genetic
20000	0.1425	0.2413	1.355	0.1583
30000	0.1787	0.3728	3.047	0.2151
40000	0.2841	0.4803	7.625	0.3012
50000	0.2449	0.6493	17.418	0.3452
60000	0.2843	0.7638	31.131	0.4416

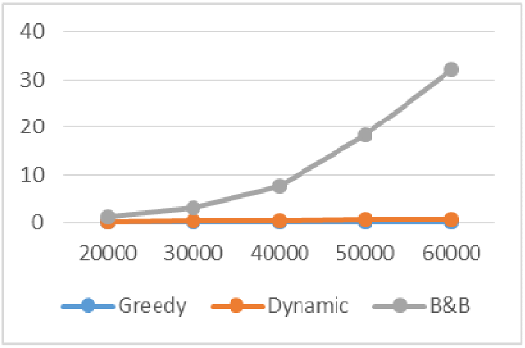


Fig. 2. Running time for Genetic, Dynamic and Greedy.

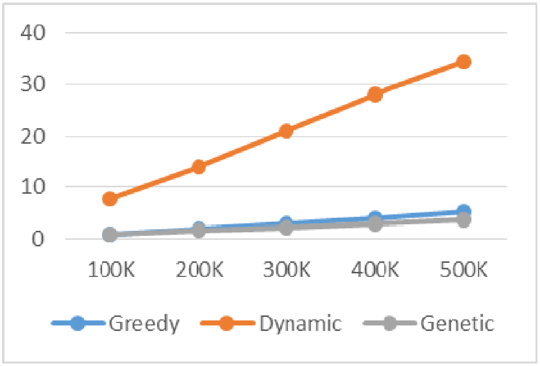


Fig. 3. Running time for B&B, Greedy and Dynamic algorithms.

TABLE VII
GREEDY ALGORITHM VS GENETIC ALGORITHM

Data Size	Greedy	Dynamic	Genetic
100	225	240	240
200	329	329	335
300	418	417	418
400	478	478	478
500	511	511	511
600	530	545	545
700	556	556	564
800	556	556	564
900	573	573	582
1000	573	573	582

TABLE VIII
GREEDY ALGORITHM VS GENETIC ALGORITHM

Data Size	Greedy	Dynamic	Genetic
100	380	388	387
200	497	511	512
300	683	684	682
400	762	774	774
500	806	818	818
600	882	889	889
700	953	947	953
800	953	953	953
900	1004	1004	1004
1000	1010	1010	1018

VII. CONCLUSIONS

All four algorithms discussed in above sections have been depicted and presented thoroughly. The overall evaluation and contrasting have been shown and all the outcome of the experiment over 0/1 knapsack problem have been discussed and demonstrated. Here The top most metric to check the effective ness of Greedy Algorithm that makes visible the algorithm in status of its running time. At this stage we could observe that performance of B&B and DP algorithms is much better than genetic and greedy algorithm in term of the all values generated by them.

Here we focus on the greedy and genetic algorithm for finding the most efficient result in favor of execution time. After performing this experiment it could be depicted that genetic algorithms achieve higher effects in phrases of how near the impacts belongs to the genuine authentic ones.

This circumstances arises due to the above that algorithms permit for multifariousness in giving choice results and they assess the fitness of these options at all steps. There are two major elements that impresses the precision of genetic algorithm. Firstly, the hypothesis of showing the problem in a way that is worthy for genetic algorithms valuation and secondly the precision of the fitness function planned for the given problem. This paper we enlighten the 0/1 Knapsack issue.

This issue well corresponded to the genetic algorithm. This may be more improved and accurate if the parameters like crossover chromosomes, mutation, and other population features etc.. can be assumed under the experiment.

The algorithm that abided the worst execution time is B&B because the complexity of the B&B moves exponentially. Although whenever we change the size of knapsack bag above the items still the performance time required by the dynamic algorithm greater is than the greedy algorithm.

REFERENCES

[1] Gossett, Eric. Discreet Mathematics with Proof. New Jersey: Pearson Education Inc., 2013.

[2] Hristakeva, Maya and Dipti Shrestha. "Solving the 0/1 Knapsack Problem with Genetic Algorithms." MICS 2014 Proceedings. <www.micsymposium.org/mics_2004/Hristake.pdf>.

[3] S. Mohanty, R. Satapathy, “An evolutionary multiobjective genetic algorithm to solve 0/1 Knapsack Problem,” IEEE Transl.

[4] Beijing, vol. 2, pp. 397–399, August 2009.P.KOLESAR, “A branch and bound algorithm for the knapsack problem. Manage,” Sci, pp. 723-735, May 2018.

[5] Adams, E. Balas, D. Zawack. The Shifting Bottleneck Procedure for Job-Shop Scheduling. Management Science, 34, 3, 391–401, .2017.

[6] Y. Yang, R.L. Bulfin. An exact algorithm for the Knapsack Problem with Setup. Int. J. Operational Research, 5, 280–291, 2018.

[7] George B. Dantzig, Discrete-Variable Extremum Problems, Operations Research Vol. 5, No. 2, April 1957, pp. 266–288,doi:10.1287/opre.5.2.266.

[8] Different Approaches to Solve the 0/1 Knapsack Problem. Maya Hristakeva, Dipti Shrestha; Simpson College.

[9] Rahman K. and Ahmed S. Performance Analysis of Genetic Algorithm for Solving the Multiple-Choice Multi-Dimensional Knapsack Problem. International Journal of Recent Trends in Engineering, 2009, vol. 2, no. 2.2017.

[10] Chen Lin, “A Heuristic Genetic Algorithm Based on Schema Replacement for 0-1 knapsack Problem”, Fourth International Conference on Genetic and Evolutionary Computing 2015

[11] Harish G , A hybrid PSO – GA algorithm for constrained optimization problems Applied Mathematics and Computation (Atlanta, USA: Elsevier) 274 292 – 305,2016.

[12] Cormen T H 2009 Introduction to algorithms (third edition) (MIT press).