A Survey Paper on Greedy Algorithms

Sakshi Dubey

University of Florida

Abstract — We discuss the Greedy algorithms as a solution to get the optimum results for a problem. This survey paper also throws light on some of the common greedy approaches and their benefits over other existing approaches.

Index Terms— Greedy Algorithms, Knapsack Problems, Graph Coloring

I. INTRODUCTION

The present era has raised to a great height in terms of advanced technologies and hence, there is a vast enhancement in the computational problem solving approach. In the ocean of thousands of efficient problem solving methods, our hunt starts for an optimum solution. Though there are various popular methods like Dynamic Programming, Divide and Conquer, Hash Tables, etc which works on sub parts to ensure us of a better solution but we are more interested in a method which is heavily reliable on the functioning of the sub parts. This lead us with an idea to discuss 'Greedy Algorithms' in this survey paper.

II. GREEDY ALGORITHMS

Most of the problem statements aim for an optimum solution. As discussed above, a preferable approach to solve tough problems is to break down in sub parts and handle each sub part one by one. In the case of Greedy Algorithms, solving sub part is the soul of whole problem set. We consider a sub problem as an individual problem and figure out the best optimum solution for that sub part without considering if it's actually a best fit for the whole problem set. This algorithm is more inclined towards a 'momentary' solution. Though we chase for an optimum solution for every sub part in sequence, yet in most of the cases it turns out to be the optimum solution for the whole problem set as well. This is the beauty of the Greedy Algorithms.

In this paper we will discuss about the critical elements of the greedy strategy & how it is implemented step by step. This paper also discusses some of the well known Greedy Algorithms and its common basics like 'Activity Selection' in a problem. This paper also covers an insight of the recent developments and contributions by different researchers in the field of Greedy Approaches and the extent to which they are successful in dealing with real world problems. Apart from the contributions of this approach to practical problems, we will also discuss some criticisms of Greedy Algorithms & scope for some future work.

III. ELEMENTS OF THE GREEDY STRATEGY

The Greedy algorithm revolves around a sequence of choices which is made at every step to aim for an optimum solution. The algorithm goes for an option that seems to be a best fit at each decision point. We follow the following generalized sequence steps to implement the Greedy Algorithm:

i. Figure out the problem set and divide the problem set in optimal substructures

ii. Determine a recursive solution for distinguished problem

iii. Visualize that there should be only one-sub problem remaining if we have chosen greedy choices in previous sub parts

iv. Draft a recursive algorithm for greedy strategy for the whole problem set

v. Transform the recursive algorithm into an iterative algorithm for the whole problem set

There are various Greedy Algorithms which we commonly use to do the computations. We will briefly discus the following popular algorithms:

A. Activity-Selection Problem
B. Kruskal's Minimum Spanning Tree Algorithm
C. Prim's MST for Adjacency List Representation
D. Dijkstra's Shortest Path Algorithm
E. Fractional Knapsack
F. Graph Coloring Algorithm

A. ACTIVITY-SELECTION PROBLEM

One of the most basic of Greedy Algorithms is the 'Activity-Selection' Problem. This problem deals with the appropriate matching of available resources with the proposed activities. Let's assume we have a set X ={x1, x2,....xn} as n activities and every activity is accomplished with a starting and finishing time si and fi respectively such that they follow this relation 0 <= si <fi< Infinity. Consider the following example to get a better understanding to filter out activities according to their starting and finishing time.

Activities schedule

I	1	2	3	4	5	6	7	8	9	10	11
$s[i]$	1	3	0	5	3	5	6	8	8	2	12
$f[i]$	4	5	6	7	8	9	10	11	12	13	14

Figure1: Activity Schedule with S[i] and f[i] values [Reference 6]

In this case, our target is to cover most of the activities such that their starting time & finishing time shouldn't overlap. This is achieved by focusing on the present sub set and choosing the maximum-size subset of compatible activities. Initially, to handle this problem brainstorming is done with the help of dynamic programming. Then we will the realize the need to focus on only one suitable choice, the greedy choice, and then we go for one that choice with only one sub problem remaining. This is how Activity-Selection problem works with the help of Greedy Algorithm.

B. KRUSKAL'S MINIMUM SPANNING TREE ALGORITHM

A Minimum Spanning tree (MST) is defined in terms of the weight associated with each graph. The graph is weighted, connected and undirected and its weight is the sum of weights of each edge. In order to solve this problem, first we sort the edges in increasing order according to their weight to figure out the smallest edge. Now make a cross check if it anyway forms a cycle. If there is cycle then forget this edge otherwise include this edge. Finally, iteratively repeat the last step until the remaining edges are just one less than its vertices.

Complexity - O(E logE)
where E is the number of edges

C. PRIM'S MST FOR ADJANCENCY LIST REPRENTATION

The working of Prim's algorithm is almost similar to the Kruskal's Algorithm. In this algorithm, we use Min Heap as priority queues to quantify the minimum edge weight. Initially we create a min heap of the size of its vertices considering first vertex as its root and allotting key values to it. Furthermore, when there is no space in the min heap then extract the minimum value node from the min heap and name it as u. Finally evaluate for every adjacent vertex v and check if it still lies in the min heap. Perform the updating of key value as u-v if it happen that the key value is more than the weight of u-v. In this way, we repeat these iterative steps until and unless min heap gets void.

Complexity - O(E LogV)

where E and V are the number of Edges and Vertices respectively

D. DIJKSTRA'S SHORTEST PATH ALGORITHM

In Dijkstra's algorithm, we prepare a shortest path tree considering root as the given source. To implement this algorithm, we prepare two list out of which one contains the used vertices & another contains the vertices that don't make the shortest path tree. Our aim is to hunt for a vertex in the other set which has least distance from the source. We always assign a distance value to all vertices in the initial graph. After every iteration we tend to update the minimum edges value. Finally, we get a non cyclic Shortest Path Tree as an output.

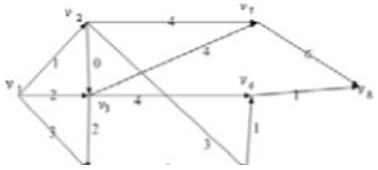

Figure 3: A weighted graph to implement Dijkstra's Algorithm

R	V_1	V_2	V_3	V_4
1	[0]	$1/v_1$	$2/v_1$	$3/v_1$
2		[1]$/v_1$	$1/v_2$	$3/v_1$
3			[1]$/v_2$	$3/v_1.v_2$
4				[3]$/v_1.v_3$
5				
6				
7				

Table 1: Depicts the value of V at every step [Reference 13]

The above shown figure gives an example of implementation of Dijkstra's shortest path algorithm step by step.

Complexity - O(E Log V)

where E and V are the edges and vertices respectively

E. KNAPSACK PROBLEM

The Knapsack Problem deals with the balance between the weight and values of certain articles. According to the problem set, we have to trade off between weight and value. There are two types of existing Knapsack problem, one is 0/1 Knapsack in which we have to either pick or leave the article. Another one is Fractional Knapsack in which one can decide how much fraction of the quantity is the best fit to the problem. A naive brute- force approach would be to try all the possible combinations and examine them one by one. On the other hand, Greedy Algorithm suggests to calculate the value to weight ratio of each item and perform the sorting according to this ratio. Now consider taking the maximum ratio and keep on adding the whole articles until you have a dearth of whole articles. In this case, fraction comes into play and thus choose the maximum fraction of the article. This is one of the ideal approach to gain the optimized solution to these kind of problems.

Complexity: O(nW)

where n is the number of items and W is the weight restriction

F. GRAPH COLORING

Graph Coloring Algorithm is widely used in day to day life for various reason like scheduling a timetable, allotting radio frequencies and drawing air plane routes, etc. Graph Coloring algorithm works on the factor of Chromatic number which is allotted to the vertices with the condition that no two adjacent numbers should share some chromatic number. Our objective is to minimize the chromatic number while keeping the condition same. This can be visualized in a way that we need minimum number of runways so that all the planes can take off in a certain duration. In this case consider a connected graph and randomly allot a color to one of its vertices. Now pick another color and allot this color to a vertex which is not connected with the first vertex. We have to iteratively perform this operation with a check that no previous vertex color should lie next to it. The number of colors required is termed as chromatic number. This greedy technique can also be used to color the graph edges and graph areas. Figure 3 gives an idea how the graph coloring is done.

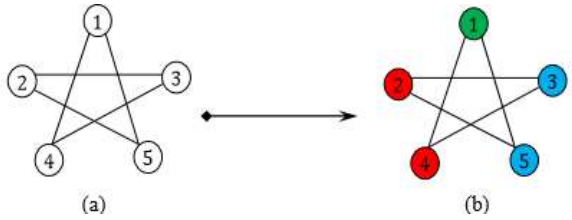

Figure 3: Graph coloring using Greedy Technique [Reference 10]

IV. RECENT CONTRIBUTION TO GREEDY ALGORITHMS IN REAL WORLD PROBLEMS

In the present decade, lot of organizations have been set up to research on the efficiency of Greedy Algorithms. A lot of real-world problems have been solved using Greedy Methods. *D.L. Donoho* has researched about the efficiency of greedy algorithms as compared to redundant dictionaries in Hilbert spaces.[Reference 3] Another interesting paper was ' Web Content Adaption for Mobile Devices: A Greedy approach' [Reference 14]. They have discussed how they are using greedy methods to make quick changes in the front end. Greedy Approach has also been covered in 'Research on Improved Greedy Algorithm for Train Rescheduling' with a new perspective. One of a real world problem of

permuted countries is depicted below in figure. In the following figure, it is visualized how graph coloring approach is used in permuting different combinations of countries.

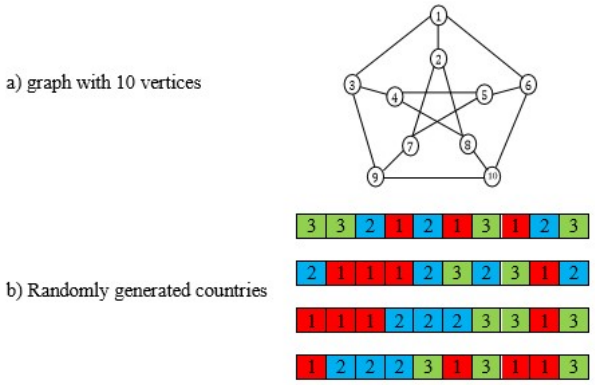

Figure 4: An example graph and created random permuted countries [Reference 10]

Another paper with the title 'Shortest Path Algorithms: An Evaluation Using Real Road Networks' [Reference 12] discusses the implementation of Greedy in developing transportation.The Application of Greedy Algorithm in Real Life [Reference 6] also discusses some real-world problems in the context of Greedy Algorithms.

V. CRITICISMS & FUTURE WORK

In this ocean of algorithms, Greedy fortunately works best in most of the cases but not in all the cases. There are various counter examples in which optimal solution for the sub parts may not result as the best solution of the problem set. In fact, in extreme cases it may result in the worst case solution. Many experimental and theoretical results of research papers and scholarly articles questions the authenticity of Greedy & favors the fact that Greedy can be destructive in extreme cases. For instance, the authors of 'When the Greedy algorithm fails' [reference 2] have researched on the asymmetric TSP that always produce a tour, which doesn't go with the optimal solution. The popular 'Travelling Salesman problem sometimes also contradict with the solutions gained by Greedy approach.

Apart from Greedy Algorithms, researchers also prefer Dynamic programming, Randomized Algorithms & Approximation Algorithms to solve the problems over Greedy algorithm. In nutshell, Greedy Algorithms is still one of a good choice for optimal solutions but it has a long road ahead to consider it as a best possible solution.

VI. CONCLUSION

Greedy Algorithms relate to multiple daily life problems. Its main objective is to optimize the current sub part at the moment which will automatically optimize the whole problem set. This survey paper concludes that though Greedy Algorithms may result in non optimized solutions in a few cases yet it is widely accepted & it has been extensively used to compute the problem sets.

REFERENCES

[1] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms (3rd ed.). McGraw-Hill Higher Education*. 2001

[2] Jorgen Bang-Jensen, Gregory Gutin, Anders Yeo. When the greedy Algorithm Fails. *Discrete Optimization*. 2004

[3] Vlamdir N. Temlyakov, Pavel Zheltov .On performance of Greedy Algorithms'. *Journal of Approximation Theory 163*(2011) 1134-1145

[4] Arogundade O.T., Sobowale B., Akinwale A.T. 2011. Prim Algorithm Approach to Improving Local Access Network in Rural Areas. *International Journal of Computer Theory and Engineering, Vol 3, No. 3, June* 2011

[5]Yingying Tian, Jianhui Lv, Liang Zheng. An Algorithm of 0-1 Knapsack Problem Based on Economic Model. *Journal of Applied Mathematics and Physics,*2013,1,31-35

[6] Jun Liu, Chuan-Cheng Zhao, Zhi-Guo Ren. The Application of Greedy Algorithm in Real Life. *International Conference on Manufacturing and Energy Engineering*. 2016

[7] Kruskal.J. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, Volume 7, pp. 48-50, 1956.

[8] Prim.R. Shortest connection networks and some generalizations. *Bell System Technical Journal*, Volume 36, pp. 1389-1401, 1957

[9] *Halldórsson, M. M*. A still better performance guarantee for approximate graph coloring*, Information*

*Processing Letters,*1993 ***45****: 19–23,*

[10] Emami H.,Lotfi S."Graph Colouring Problem Based On Discrete Imperialist Competitive Algoritm'

[11] James B. Orlin, Kamesh Madduri, K. Subramani, and M. Williamson. . A faster algorithm for the single source shortest path problem with few distinct positive lengths. *J. of Discrete Algorithms 8, 2* (June 2010), 189-198

[12] Zhan, F. Benjamin; Noon, Charles E.. "Shortest Path Algorithms: An Evaluation Using Real Road Networks". *Transportation Science* ***32*** *(1): 65–73.*February 1998

[13] Shu-Xi W. The Improved Dijkstra's Shortest Path Algorithm and Its Application. *International Workshop on Information and Electronics Engineering Procedia Engineering 1186-1190*. 2012

[14]Cserkuti P.,Szabo Z.,Eppel T.,Pal J. SmartWeb-Web Content Adaption for Mobile Devices. Muegyetem rkp. 3-9