Hamiltonian cycle and TSP: A backtracking approach

[1]Dipak Patel,[2]Nishant Doshi

Sardar Vallabhbhai National Institute of Technology
Surat,Gujarat,India

[3]Shakti Patel, [4]Dishant Soni

Sankalchand Patel College of Engineering
Visnagar, Gujarat, India

Abstract—Backtracking is one of the strategies to reduce the complexity of a problem. Backtracking mainly useful when there is a no solution by going forward in that direction so we required backtracking from it to reduce the complexity and save the time. Backtracking has ability to give same result in far fewer attempts than the exhaustive or brute force method trials. This paper gives the recursive algorithm for Hamiltonian cycle and TSP (travelling salesman problem) based on the backtracking approach. If at any stage it is detected that the particular input or combination will not lead to an optimal solution than we can discard and a new input can be selected.

Keywords- Algorithm; Backtrackin; Graph; Hamiltonian path; TSP.

I. INTRODUCTION

The Travelling salesperson problem is one of the problem in mathematics and computer science which had drown attention as it is easy to understand and difficult to solve. A Hamiltonian cycle of a directed graph G = (V, E) is a cycle that contains each vertex in V once. Backtracking is useful in the case of travelling salesman problem in a sense that assumes the number of cities is 10. The brute force solution will require about 10! trials. If we can use backtrack and in the halfway through tour that this is not going to be an optimal tour than we discard it saving the trials. Till now there are many papers published. The history of TSP is given in hoffman and wolfe [2]. The importance of TSP useful as it is the representative of a class of problem known as NP-complete. If one can find an efficient algorithm i.e. guaranteed to find an optimal solution in polynomial time [1]. If we can find the solution for TSP than all the problems in class NP can be solved in polynomial time. Till now no one had found polynomial algorithm. There are papers [3-10] which discuss about approximation of the TSP problem. In [11] the applications of the TSP were discussed. In this paper we had outlined the approach based on backtracking method.

II. BACKTRACKING APPROACH

As shown in fig 1 the graph G (V, E) is given containing N=4 vertices and 6 edges, here N is the total number of vertices in a graph. As shown in the fig 1 the four vertices A, B, C and D are given. We can consider them as different cities. The distance between A and B is 4. The distance between A and C is 3 and so on. Here we using undirected weighted graph in form of a matrix where row and columns for particular cities. In the same way we can use the directed graph with weighted length. Here we assume that all the weight length is non-negative as they represent the distance between two cities. And later this notion of backtracking is applied to TSP problem.

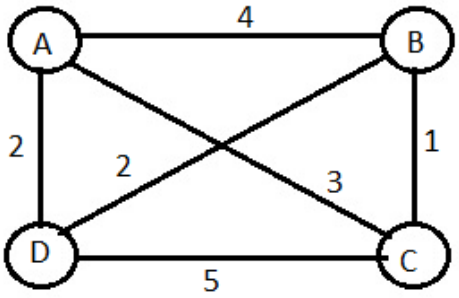

Fig. 1.

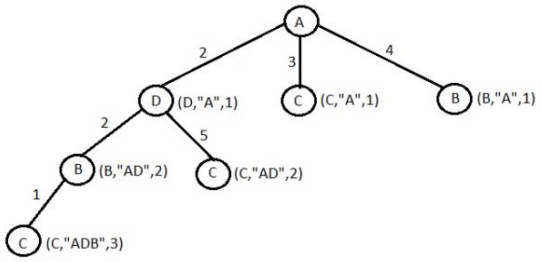

Fig. 2.

Assume that tour start from vertex A. In fig 2 the tuple notation (X, Y, Z), where X is current node, Y is string containing ancestors of node X and Z is the total number of nodes we had encountered till present. When Z=N-1 and E(X, A) is true, we finished. In this example there is no any backtracking. As shown in the fig 2 there is total 3 edges outgoing from vertex A. Here we considering the minimum weight out of them but in general we can consider in the ordering they come in the adjacency matrix. Now we start from vertex D and make a tuple notation. And then go for the edges here we cannot consider edge between D to A because A is the parent or one of the ancestors of the node D. now we start from vertex B and as A and D are the ancestors for node B so only one edge remaining i.e. C. And at node C we had total edges are 3 and there is edge between nodes C to node A. so we found the Hamiltonian cycle. If we apply the brute force method than in that case first we have to generate the entire N! permutation of the nodes and then it is required to test each permutation whether it is a Hamiltonian cycle or not. So in the backtracking, time will be less and algorithm can be efficient.

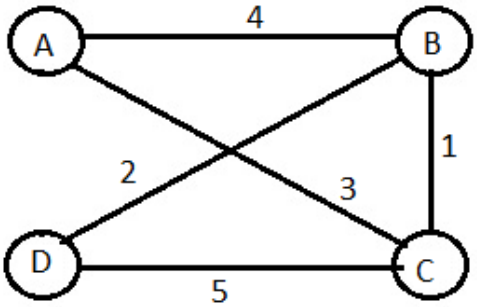

Fig. 3.

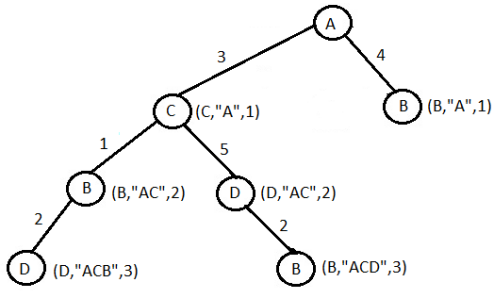

Fig. 4.

Now consider the fig 3. As shown in fig 3 the minor change is that the edge between nodes A to node D is removed and the edge weight are remained to be same as in fig 1. Now in this example of graph we can see the use of backtracking and see that how it is useful for determine the Hamiltonian cycle. As shown in fig 4, we will start from vertex as assuming that it is the starting vertex. Now there is two edges out of one is to node C and the other is to node B. so we can start from any vertex. Let's start from vertex C. in algorithm later, we consider the vertex first which come in the matrix notation first. As a recursive call we reach at node D, but we don't have E (D, A) exist so we can have Hamiltonian path but not a Hamiltonian cycle. So we had to backtrack to B, now B don't have any edge remaining, so again backtrack to C and continued with child node D. Then A-C-D-B-A is the HC. There may be many HC possible in a given graph, the minimal of them is the travelling salesman problem. In fig 4 to find a HC we actually finding the shortest path between C to B containing all the remaining vertices except A. we can reduce the algorithm complexity by adding the logic, to start the tour from the vertex which has out-degree is minimum. Then in fig 4 no need to explore node B (as a child of node C) further because we know that the path must lead to node B and from it to A. so the moment we got node B as a child node of node C we not need to go further as that will not lead to a Hamiltonian cycle. Now we develop the algorithm for the Hamiltonian cycle and the tsp. in every recursive algorithm there is required the base step to stop the recursive thing gone further. So when *edges=N* and there must be edge between the node and the starting node so we can write the string that we had as that is the path and if one want the length than it can also be calculated. So the code will be.

```
 If (edges=N and E (SN, Node))
Then
    Print (String) //HC found
```

```
    Exit
```

Now to prevent infinite call and as a precaution we take the condition that if edges=N the moment we stop and return from that point so the code can be.

```
If (edges=N)
Then
    Return
```

Now the thing is we have to create each child of the node and check if that child is in the ancestors in that node if yes than discard that child or edge to that node.

```
For each child X of Node and NotIn(X, String)
```

Now if the child not is not in the ancestors list than we have to add that node in the string which contain the list of ancestors and then call the recursive node containing the child node the new string containing the child node and makes edges increment by one as follows. Here we assume the name of the function or algorithm is HC.

```
For each child X of Node and NotIn(X, String)
Do
    String = String + info(X)
    HC(X, String, edges+1)
Done
```

So based on the above discussion we can write the complete algorithm for Hamiltonian cycle based on the backtracking as follow.

```
 Algorithm HC (Node, String, edges)
 {
1.  If (edges=N and E(SN,Node))
2.  Then
3.      Print (String) //HC found
4.      Exit
5.  If (edges=N)
6.  Then
7.      Return
8.  For each child X of Node and NotIn(X,String)
9.  Do
10.     String = String + info(X)
11.     HC(X,String, edges+1)
12. Done
13. Return
 }
```

Now we will look at the problem of TSP from the Hamiltonian cycle problem. In a given weighted graph there are many Hamiltonian cycle can be possible but out of which the minimum length one the TSP. The first step is the base condition or when we stop in the recursive algorithm. Here in this case we have to examine each node and every edge and every possible combination of it. Here what we do when we get the Hamiltonian cycle. Than we have to compute the length of that Hamiltonian cycle and compare with the previously minimum calculated Hamiltonian cycle length if the newer is the minimum than we can save the string and the length of it as a prec_length as shown in the following code.

```
If (edges=N and E (SN, Node))
Then
    If (length < prev_length)
    Then
        prev_length=length
        prev_string=string
        Exit
    Return
```

Now once in the in between the moment we got the path length is greater than the previously calculated

Hamiltonian cycle path length. Then there is no any meaning to go or explore that path as we definitely get the not less than the minimum weight. So for this one and here we get the advantage of backtracking that we not need to all the paths that contain or follows from that one so the code as follows.

```
If (length > prev_length)
Then
   Return
```

Now the next one is same as Hamiltonian cycle problem as follows.

```
For each child X of Node and NotIn(X, String)
Do
   String = String + info(X)
   TSP(X,String,edges+1,length+E(Node,X), prev_string,prev_length)
Done
```

```
Algorithm      TSP(Node,String,edges,length,prev_string,prev_length)
  {
1.  If (edges=N and E(SN,Node))
2.  Then
3.        If (length < prev_length)
4.        Then
5.             prev_length=length
6.             prev_string=string
7.             Exit
8.        Return
9.  If ( length > prev_length )
10. Then
11.       Return
12. For each child X of Node and NotIn(X,String)
13. Do
14.       String = String + info(X)
15.       TSP(X,String,edges+1,length+E(Node,X), prev_string,prev_length)
16. Done
17. Return
  }
```

III. CONCLUSION

This paper discussed the simple backtracking approach to solve the TSP and Hamiltonian cycle. Discussions show that in worst case it's come to non-polynomial behavior. The advantage of this approach is, if there is an optimal solution than it must be found. So no any approximation answer will be return. In future if we can device more efficient method to solve the TSP than we can say P=NP.

REFERENCES

[1] T.H.Cormen, C.E. Leiserson, R. L. Rivest, C. Stein, "Introduction to algorithms", second edition, McGraw-Hill publication, 2002.

[2] J. Hoffman and P. Wolfe (1985), "History" in The Traveling Salesman Problem, Lawler, Lenstra, Rinooy Kan and Shmoys, eds., Wiley, 1-16.

[3] Vijay V. Vazirani, Approximation algorithms, Springer-Verlag New York, Inc., New York, NY, 2001

[4] Piotr Berman , Marek Karpinski, 8/7-approximation algorithm for (1,2)-TSP, Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, p.641-648, January 22-26, 2006, Miami, Florida

[5] Bläser, M. 2004. A 3/4-approximation algorithm for MaxATSP with weights zero and one. In Proceedings of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX). Lecture Notes in Computer Science, vol. 3122. 61--71.

[6] Hans-Joachim Böckenhauer , Juraj Hromkovic , Ralf Klasing , Sebastian Seibert , Walter Unger, An Improved Lower Bound on the Approximability of Metric TSP and Approximation Algorithms for the TSP with Sharpened Triangle Inequality, Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science, p.382-394, February 01, 2000

[7] Christofides, N. 1976. Worst-case analysis of a new heuristic for the travelling salesman problem. In Algorithms and Complexity: New Directions and Recent Results, J. F. Traub, Ed. Academic Press, 441.

[8] Haim Kaplan , Moshe Lewenstein , Nira Shafrir , Maxim Sviridenko, Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs, Journal of the ACM (JACM), v.52 n.4, p.602-626, July 2005

[9] Moshe Lewenstein , Maxim Sviridenko, A 5/8 Approximation Algorithm for the Maximum Asymmetric TSP, SIAM Journal on Discrete Mathematics, v.17 n.2, p.237-248, 2004

[10] Christos H. Papadimitriou , Mihalis Yannakakis, The traveling salesman problem with distances one and two, Mathematics of Operations Research, v.18 n.1, p.1-11, Feb. 1993
[11] PUNNEN, A.P.: 'The Traveling Sales-man Problem: Applications, Formulations and Variations', in G. Gutin and A.P. Punnen (eds.): The Traveling Salesman Problem and its Variations, Kluwer, Dordrecht, 2002.