Constructing the Graph and Finding the Branches uniquely using the Cycle Space

Kritika Sharma*, Saurabh Singh†, Anujraj Goel‡
Indian Institute of Information Technology, Allahabad
*icm2014502@iiita.ac.in
†iwm2014004@iiita.ac.in
‡iim2014002@iiita.ac.in

***Abstract*—The problem of finding the branches of a graph using the cycle space is having its value in various fields. The problem is that given a Cycle space, we are required to construct the graph and also find the branches uniquely. In this paper, we proposed a simple algorithm for constructing the graph using cycle space and then to find the branches with respect to all the spanning trees of the graph constructed. The complete algorithm has exponential time complexity of O(Log(V)*2^E)**

***Index Terms*—spanning subgraph . eulerian . cycle space . branch**

I. Introduction

In graph theory, a spanning subgraph of G(V,E) is a graph g(V',E') where $|E'| <= |E|$ and E' can contain any number of edges, even zero edges. An eulerian subgraph of G(V,E) is any subgraph g(V',E') where $|V'| <= |V|$ and $|E| <= |E'|$ such that every vertex in V' has even degree. Now, a cycle space [3] is the collection of all the eulerian spanning subgraphs [4] in the graph. A branch is simply any edge present in the spanning tree of a graph.

In fig.1, a graph G(V,E) is given. For this graph, two of the eulerian spanning subgraphs is as given in fig.2. For this graph, we will have in total 23 spanning trees as can be found using Kirchoff's Theorem [2]. Some of these spanning trees are as shown in fig.3.

In this paper, we first deal with the construction of graph when given the cycle space as input. Also, after the construction of the graph, we find the branches uniquely with respect to each possible spanning tree [1] of that graph. For this we find all the possible spanning trees of the constructed graph. Now, as we know that the construction of some specific graphs are not unique (graphs with pendant vertices). So, in this paper we assumed that the input graph does not a have a pendant vertex (i.e. degree of every vertex is greater than or equal to 2).

II. Motivation

This paper is motivated by the various real life examples where eulerian subgraphs are important and information about the structures are given in the form of collection of eulerian spanning subgraphs only. We can use this information only to solve various problems. Navigating a trip to see all different regions of the world. In Floor Designs, Fly the friendly skies and pretending You are the pilot. They can also be used to by mail carriers who want to have a route where they don't retrace any of their previous steps.It also has applications in Sequencing of DNA. They are also useful to painters.

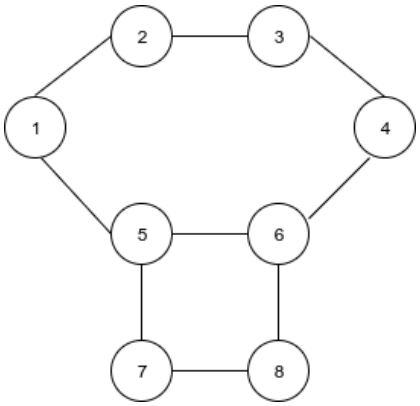

Fig. 1. A graph - G(V,E)

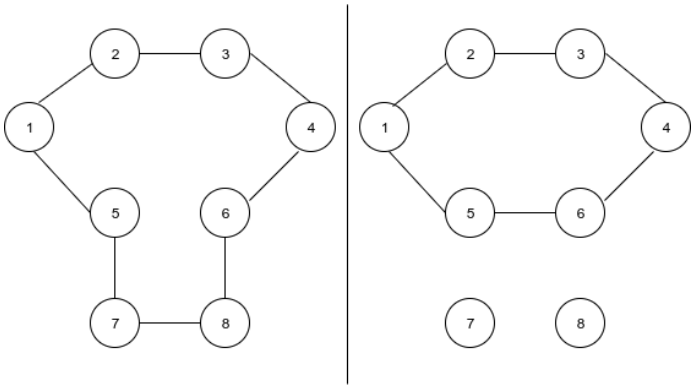

Fig. 2. Two eulerian spanning subgraphs of G(V,E)

III. Methods and Descriptions

In this paper, we divided the problem into two parts. In the first part, the motive is to construct the graph with the knowledge of the given cycle space. In the second part, we present an algorithm to find all the possible spanning trees so that branches with respect to each can be uniquely found.

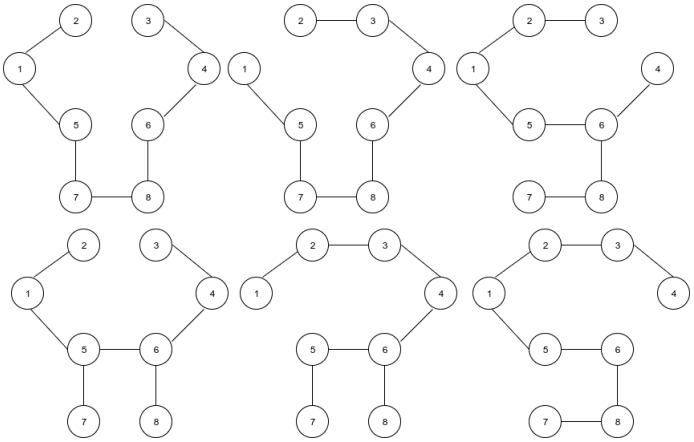

Fig. 3. Some of the spanning trees of G(V,E)

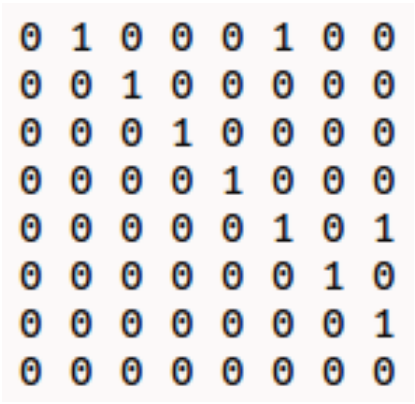

Fig. 4. Adjacency matrix of G(V,E)

For constructing the graph we simply traverse through all the eulerian spanning trees and we insert all the edges of each of them into the graph if not already inserted and at last, we represent this graph with the help of an adjacency matrix. For e.g. using the cycle space given for graph in fig. 1, we constructed the original graph. Its adjacency matrix is given in fig.4. After the construction of the graph, we find all the possible spanning trees and for each of these we simply mention the branches. For finding all the possible spanning trees, we used an exponential algorithm as described. Some of these are given in fig.3.

IV. Implementation and Results

The algorithm 1 given below is for constructing graph. We simply consider all the edges in each subraph of cycle space. If that edge is not already considered we include it in the graph. Similarly, we include the vertices.

The algorithm 2 presented below finds all the possible spanning trees using recursion. We check if the edge forms the cycle with the current subset of edges taken. If cycle is there we don't take edge. Else we take the edge and apply recursion until all the vertices are covered.

The algorithm suggested in this paper is exponential with time complexity of O(Log(V)*2^E). The number of vertices vs time taken graph is as shown in fig. 4 and fig.5. Time is increasing exponentially as expected. The number of nodes with their respective time taken is also tabulated as in table 1.

Algorithm 1 Graph Construction

procedure CONSTRUCT(C) ▷ where C = Cycle Space
  $V \leftarrow$ `empty set`
  $E \leftarrow$ `empty set`
  for `each sub in C` **do**
    for `(each edge in sub)` **do**
      $x \leftarrow edge.source$
      $y \leftarrow edge.destination$
      if $x > y$ **then**
        $swap(x, y)$
      if *edge between x and y not considered* **then** $E = E \cup edge(x, y)$
      if *x not considered* **then** $V = V \cup x$
      if *y not considered* **then** $V = V \cup y$
  return (V, E)

Algorithm 2 Spanning Trees

procedure SPANNINGTREES(SP,EE,IND,E)
  if *sp not already considered* **then**
    $Print(sp)$
  if *ind is greater than equal to size of E* **then**
    $Return$
  $SpanningTrees(sp, ee, ind + 1, E)$
  $x \leftarrow E[ind].source$
  $y \leftarrow E[ind].destination$
  if *x-y does not form a cycle in sp* **then** $sp = sp \cup (x - y)$
    $SpanningTrees(sp, ee - 1, ind + 1, E)$
procedure MAIN(G(V,E))
  $sp \leftarrow$ `empty set`
  $np \leftarrow$ `Number of vertices in V`
  $SpanningTrees(sp, np - 1, 0, E)$

Table 1.Time Taken

Number of Nodes	Time Taken(in ms)
3	0.117
5	0.3470
6	0.4600
7	1.3120
8	2.7910
9	3.1680
12	249.1140
13	482.5180
15	1938.6630
17	8681.6350
18	16978.010

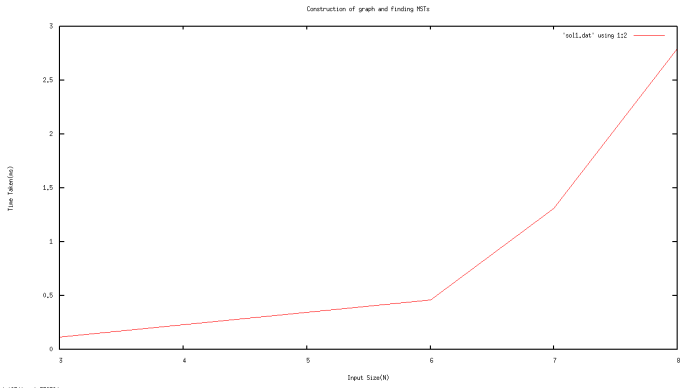

Fig. 5. Number of Nodes vs Time Taken(ms) graph

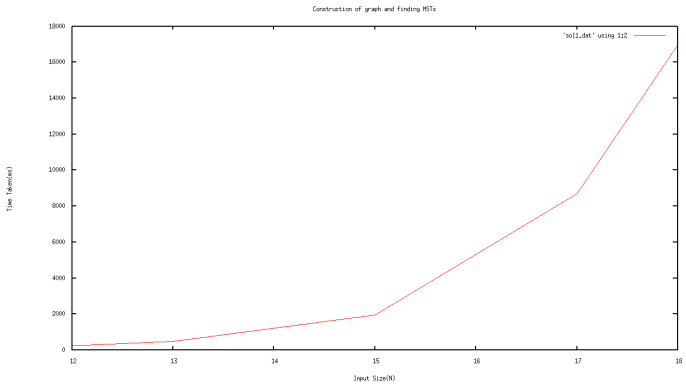

Fig. 6. Number of Nodes vs Time Taken(ms) graph

V. Conclusion

In this paper, we presented an algorithm to construct the original graph when the cycle space is given as the input. We simply traverse all the edges and vertices in each entry of cycle space and include it in the graph to be constructed if not already included. After constructing the graph, we recursively find all the possible spanning trees. This takes an exponential time with time complexity of O(Log(V)*2^E). We will further try to improvise our solution.

VI. References

[1] http://www.geeksforgeeks.org/greedy-algorithms-set-2-kruskals-minimum-spanning-tree-mst/ [Last accessed on September 27, 2017]

[2] http://www.geeksforgeeks.org/total-number-spanning-trees-graph/ [Last accessed on September 27, 2017]

[3] https://en.wikipedia.org/wiki/Cycle_space [Last accessed on September 27, 2017]

[4] https://proofwiki.org/wiki/Definition:Spanning_Subgraph [Last accessed on September 27, 2017]