Given a cycle space of a graph generate all circuits other than fundamental circuits

Akshat Aggarwal
IIM2014005
7th Semester Dual Degree (IT)

Yogesh Gupta
IRM2014004
7th Semester Dual Degree (IT)

Vishesh Middha
ISM2014007
7th Semester Dual Degree (IT)

***Abstract*—In this assignment we strive to find circuits other than the fundamental circuits in a graph.**

***Index Terms*—graph, circuits, fundamental circuits, spanning trees**

I. Introduction

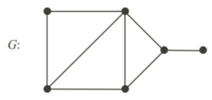
We consider an undirected graph $G = (V, E)$ where,

- V is a set of vertices (or nodes), and
- $E \subseteq (V \times V)$ is a set of edges.

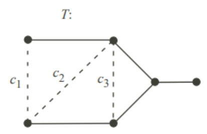
Our objective is that given a cycle space of a graph(Definition 3), find out the circuits(Definition 2) other than fundamental circuits(Definition 2) in the graph. After introducing the necessary notation and basic definitions, we explain our devised method of finding the circuits from cycle space which are not fundamental circuits.

A. Definitions and Notations

1) **Circuit** -
A closed walk in a graph in which no vertex appears more than once(except the first and the last vertex) is called as a circuit. [1].
2) **Fundamental circuit** -
For a graph G with m edges and a spanning tree T of the graph G with μ number of edges called as branches, there will be $r = m - \mu$ edges left. These are called as chords. Adding a chord to the spanning tree will definitely form one circuit. Such a circuit so formed by adding a chord to the spanning tree is called as a fundamental circuit. [2]
For example consider the graph G shown in figure. 1a with one spanning tree T shown in figure .1b. We are then left with three edges c_1, c_2, c_3 as chords. Adding each chord to the spanning tree T will lead to formation of three circuits as shown in figure. 2. These three circuits are fundamental circuits.



(a) Graph G



(b) Spanning Tree T

Fig. 1: Graph and one of its spanning tree

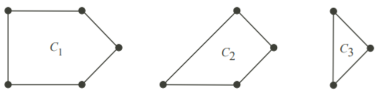


Fig. 2: Fundamental Circuits of graph G with respect to spanning tree T

3) **Cycle Space** -
The cycle space of an undirected graph is the set of its Eulerian Subgraphs [1].
Formally, let $G = (V, E)$ be a graph, then the cycle space of G, denoted by C is subset of edge-space i.e. 2^E containing ϕ, all cycles in G (where each cycle is treated as a set of edges) and all unions of edge disjoint cycles in G. For example, graph G shown in figure. 3 has the cycle space C shown in figure. 4

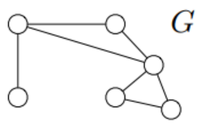


Fig. 3

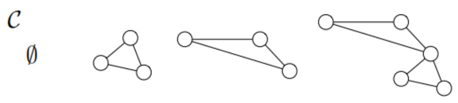


Fig. 4: Cycle space of graph G shown in 3

II. Motivation

Graphs are very important data structure, finding uses in numerous applications such as in network related algorithms, routing, finding relations among objects, shortest paths and many more real life related applications. On e-commerce websites, relationship graphs are used to show recommendations. Connecting with friends on social media, where each user is a vertex and the connection between two users is represented by an edge, is also an important application of graphs.

Locating circuits in a graph is an important topic and useful in many situations. A common example is the traveling salesman problem in which we have to find a minimum cost circuit for visiting each city exactly once. Of all circuits, fundamental circuits are particularly much more interesting since we can form all other circuits as a linear combination(i.e. ring sum) of these fundamental circuits [2]. As a result, we can find all other circuits(non-fundamental circuits) and apply different tasks such as minimum, longest and others as per our need.

III. Methods and Algorithms

Here, we do not find the cycle space of a graph and take it as an input parameter. However, a cycle space can be found out using a naive method in exponential time.
Although, the total number of fundamental circuits in a graph remains the same, the circuits which are fundamental change according to different spanning trees. A circuit which is fundamental with respect to one spanning tree may not remain fundamental with respect to some other spanning tree [2]. So, for our purpose we consider any one spanning tree of the graph which be found in linear time using an algorithm such as Breadth-first or Depth-first search.

First, we find out the fundamental circuits of the graph with respect to a spanning tree. For this, we follow a straight-forward approach consisting of the following steps -

1) Separate the edges of the graph into branches(those that are part of the spanning tree) and chords(those edges that are not part of the spanning tree).
2) Initialize an empty list of fundamental circuits
3) Add a chord to the spanning tree
4) Find the cycle formed and add it to the list of fundamental circuits
5) Remove the chord from the spanning tree
6) Repeat steps 2 to 5 for each chord

A simple procedure for above steps is shown in Algorithm 1.

Algorithm 1 Finding fundamental circuits in a graph with respect to a spanning tree of the graph

```
Input: Graph $G$, a spanning tree $T$ of $G$
Output: List of fundamental circuits in $G$ with respect to $T$
  $m \leftarrow$ number of edges in a graph
  $\mu \leftarrow$ number of branches in spanning tree $T$
  $r \leftarrow m - \mu$ , number of chords
  for each chord $r_i$ do
    Add chord $r_i$ to the spanning tree $T$. Call this $G^{'}$
    Find the circuit using DFS starting from one of the vertices in $G^{'}$
    Save the circuit, $C_i$ found to $result$
    Remove chord $r_i$ from the spanning tree $T$
  end for
  return $result$
```

Once the fundamental circuits are found we can find these circuits in the given cycle space and remove them, to finally obtain circuits other than the fundamental circuits. Also, we remove the ϕ element from the result.

Algorithm 2 Finding circuits other than fundamental circuits

```
Input: Cycle space of the graph and list of fundamental circuits(from Algorithm 1)
Output: List of circuits other than fundamental circuits
  $result \leftarrow cyclespace$
  Remove from $result$ the null element, $\phi$ and element of all edges
  for each fundamental circuit $C_f$ do
    Remove $C_f$ from $result$
  end for
  return $result$
```

IV. Results and Conclusion

First, we found a spanning tree of the given graph. Time complexity of doing this $O(V + E)$.

Then, we found all the fundamental circuits of the graph with respect to a spanning tree. For every chord (edge which is not in the spanning tree) , we added it to the spanning tree and found the formed circuit using DFS and pushed this circuit to the fundamental circuits set. Time complexity of finding all fundamental circuits is $O(\mu \times (V - 1))$, where μ - number of chords.

After finding all the fundamental circuits we remove all the fundamental circuits from the cycle space set. Time complexity of removing these fundamental circuits is $O(X \times \mu)$, where X is size of cycle space, and μ is number of chords. It is assumed that time required for comparison is constant i.e.$O(1)$. However, using something similar to sets in C++ will lead to logarithmic time for comparisons between cycles.

Below is the plot of time complexity vs size of input :-

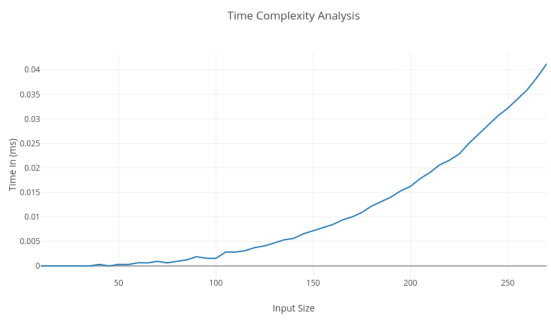


Fig. 5: Time vs input size

References

[1] Gross, Jonathan L. Yellen, Jay (2005), "4.6 Graphs and Vector Spaces", Graph Theory and Its Applications (2nd ed.), CRC Press, pp. 197–207, ISBN 9781584885054.
[2] Graph Theory with Applications to Engineering and Computer Science, Narsingh Deo.