Finding Eulerian Cycles in Directed Graphs

Investigating Hierholzer's Algorithm

Robert Michael Milliner
School of Computing
Weber State University
Ogden, Utah, USA
robertmilliner@mail.weber.edu

ABSTRACT

This paper investigates the efficiency of finding Eulerian cycles, particularly focusing on whether they can be found in linear time. Euler cycles, which traverse all edges of a graph while visiting each vertex at least once, are crucial in various applications. Hierholzer's algorithm is explored as a potential solution for this problem. The algorithm's applicability to directed graphs is examined, with particular attention to the conditions necessary for the existence of Euler cycles. Pre-processing steps include checking that the directed graph is strongly connected and that vertex in-degrees and out-degrees are equal. Hierholzer's algorithm, known for its linear time complexity, is described in detail, demonstrating its ability and efficiency to find Eulerian cycles in directed graphs. The paper concludes that, given the existence of one, Eulerian cycles can indeed be found in linear time using Hierholzer's algorithm.

KEYWORDS

Graph: "A graph G = (V, E) consists of a finite set V of vertices and a set E of edges joining different pairs of distinct vertices"**[1]**, **Directed Graph(Digraph)**: A graph whose edges are directed, indicating a specific direction from one vertex to another, **Vertex(Node)**: A point where edges meet in a graph, **Edge**: A connection between two vertices in a graph, **Directed Edge**: An edge with a direction, **In-Edge**: A directed edge going into a vertex, **Out-Edge**: A directed edge leaving a vertex, **Adjacent**: Two vertices are called adjacent if they are connected by an edge, **Incident**: Two edges are called incident, if they share a vertex, **Degree**: The number of edges incident to a particular vertex, **In-Degree**: The number of In-Edges incident to a vertex, **Out-Degree**: The number of Out-Edges incident to a vertex, **Path**: "A path P is a sequence of distinct vertices, written P = x1–x2– ⋯ –xn, with each pair of consecutive vertices in P joined by an edge"**[1]**, **Connected**: "A graph is connected if there is a path between every pair of vertices"**[1]**, **Strongly Connected**: A directed graph is strongly connected if there is a directed path between every pair of vertices in the graph, **Breadth First Search**: A graph traversal algorithm that starts at a specified vertex and explores all its neighbors at the present depth prior to moving on to the vertices at the next depth level, **Cycle**: A closed path in a graph, where the first and last vertices are the same, **Eulerian Cycle(Euler Cycle)**: "An Euler cycle of a strongly connected, directed graph is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once"**[2]**

1 Motivation and background

Having a strong background in Mathematics, and being particularly fond of the field of Combinatorics, the foundational problem of finding Euler cycles was very interesting to me. In particular I wanted to know **Can Eulerian Cycles be Found in Linear Time?** Upon further investigation I learned of Hierholzer's Algorithm (published in 1873) which claims to find Eulerian cycles in linear time.

1.1 Pre-Processing

Before Hierholzer's Algorithm can be performed on a Digraph G, it must be known whether an Euler cycle exists within G. For there to exist an Euler cycle in G, two conditions must be met;

1. The in-degree must be equal to the out-degree for every vertex in G.
2. G must be strongly connected

The first requirement is trivial, as we just check this condition for each vertex; this was implemented in this project with a time complexity of O(V), where V is the number of vertices in G.

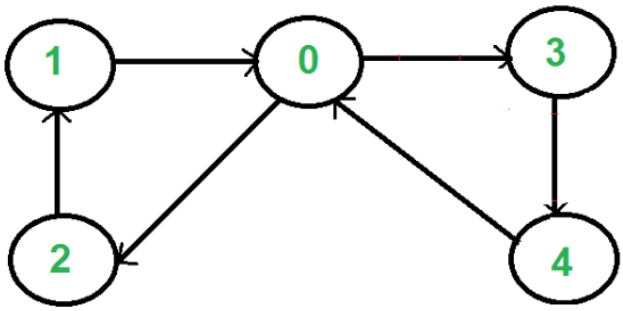
The second requirement was more difficult to test for and was implemented by running Breadth First Search V times using each vertex as the source once. The time complexity of this implementation is O(V^2 + VE), where V is the number of vertices in G, and E is the number of edges in G. There are more efficient algorithms to test for strong connectivity, but the goal of this project was to find Euler cycles, not to verify their existence.

1.2 Hierholzer's Algorithm

"The basic idea of Hierholzer's algorithm is the stepwise construction of the Eulerian cycle by connecting disjunctive cycles. It starts with a random node and then follows an arbitrary unvisited edge to a neighbor. This step is repeated until one returns to the starting node. This yields a first cycle in the graph. If this cycle covers all nodes it is an Eulerian cycle and the algorithm is finished. Otherwise, one chooses another node among the cycles' nodes with unvisited edges and constructs another cycle, called a subtour. By choice of edges in the construction the new cycle does not contain any edge of the first circle, both are disjunct. However, both cycles must intersect in at least one node by choice of the starting node of the second cycle. Therefore one can represent both cycles as one new cycle. To do so, one iterates the nodes of the first cycle and replaces the subtour's starting node by the complete node sequence of the subtour. Thus, one integrates additional cycles into the first cycle. If the extended cycle does include all edges the algorithm is finished. Otherwise, we can find another cycle to include"**[3]**.

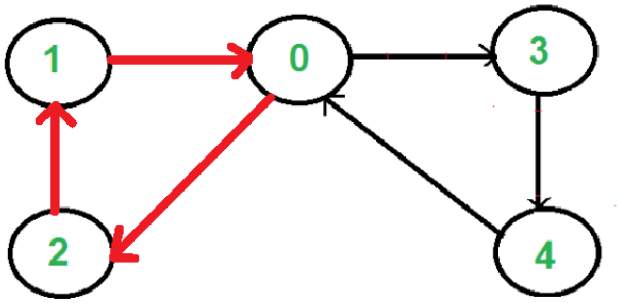
1.3 An Example

Consider the Digraph G pictured below:

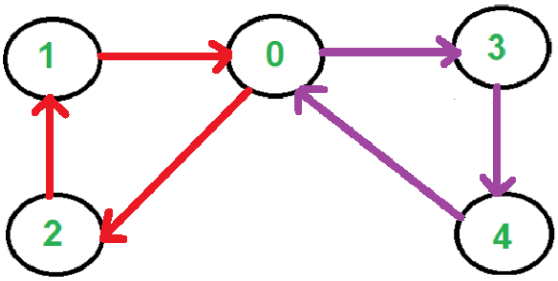


[4]

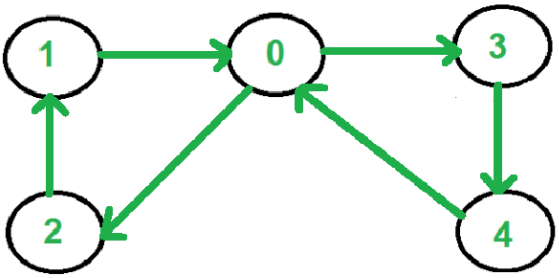
It is trivial to see that every vertex in G has its in-degree equal to its out-degree, and, by inspection the reader can verify that G is strongly connected. Therefore, an Euler cycle exists within G and Hierholzer's Algorithm can be performed. To begin the algorithm, we can arbitrarily select vertex 0 (V0) as the starting vertex, then arbitrarily choose an edge leaving V0, we will choose the edge (0,2), we then choose the only edge leaving V2, (2,1), then choose the only edge leaving V1, (1,0), because we have returned to V0 we have constructed the first disjunct cycle C1, V0-V2-V1-V0 this is shown below:



Because there is a vertex in our initial cycle with un-traversed edges, we will begin the second iteration of the algorithm. We will choose the only edge leaving V0, (0,3), we then choose the only edge leaving V3, (3,4), finally we choose the only edge leaving V4, (4,0), because we have returned to our starting vertex V0, we have found another disjoint cycle C2, V0-V3-V4-V0 this is shown below:



Because there are no vertices left in any of our disjoint cycles, we can perform the final step of our algorithm by inserting C2 into C1 anywhere where the starting vertices are the same, i.e. C1 is V0-V2-V1-V0 so we will replace the second V0 in the cycle with C2 to get V0-V2-V1-C2 thus the Euler cycle produced by Hierholzer's algorithm is: V0-V2-V1-V0-V3-V4-V0 this is shown below:



Notice that every edge in G has been visited exactly once, and every vertex in G has been visited at least once. Therefore, V0-V2-V1-V0-V3-V4-V0 is a valid Euler cycle.

1.4 Results

Because this algorithm checks every vertex, and traverses every edge exactly once, it has time complexity O(V + E) where V is the number of vertices in G and E is the number of edges in G.

1.4 Conclusion

Based on my results, my research question has been answered. Yes, Eulerian cycles can be found in linear time! In the future, I would like to refine my implementation of checking for strong connectivity, and implement this algorithm for undirected graphs. I would then have a fully robust program that could find an Euler cycle in any graph, if one exists, in linear time.

REFERENCES

[1] Tucker, Alan. Applied Combinatorics. 6th ed., 1943.

[2] Cormen, Thomas H., et al. Introduction to Algorithms. 4th ed., The MIT Press, 2022.

[3] Becker, Mark J. “Computing Eulerian Cycles.” Hierholzer’s Algorithm, algorithms.discrete.ma.tum.de/graph-algorithms/hierholzer/index_en.html.

[4] “Euler Circuit in a Directed Graph.” GeeksforGeeks, GeeksforGeeks, 31 Jan. 2023, www.geeksforgeeks.org/euler-circuit-directed-graph/.