Hindawi
Mathematical Problems in Engineering
Volume 2023, Article ID 9829831, 1 page
https://doi.org/10.1155/2023/9829831

Retraction

Retracted: Asymptotically Effective Method to Explore Euler Path in a Graph

Mathematical Problems in Engineering

Received 10 October 2023; Accepted 10 October 2023; Published 11 October 2023

Copyright © 2023 Mathematical Problems in Engineering. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Peer-review manipulation

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] M. Fahad, S. Ali, M. Khan, M. Husnain, Z. Shafi, and A. Samad, "Asymptotically Effective Method to Explore Euler Path in a Graph," *Mathematical Problems in Engineering*, vol. 2021, Article ID 8018373, 7 pages, 2021.

Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 8018373, 7 pages
https://doi.org/10.1155/2021/8018373

RETRACTED

Research Article

Asymptotically Effective Method to Explore Euler Path in a Graph

Muhammad Fahad,[1] **Sikandar Ali**,[2] **Mukhtaj Khan**,[2] **Mujtaba Husnain**,[3] **Zeeshan Shafi**,[3] **and Ali Samad**[3]

[1]*Govt.College Civil Lines, Multan 66000, Pakistan*
[2]*Department of Information Technology, The University of Haripur, Haripur 22621, Khyber Pakhtunkhwa, Pakistan*
[3]*Faculty of Computing, The Islamia University of Bahawalpur, Bahawalpur 63100, Pakistan*

Correspondence should be addressed to Sikandar Ali; sikandar@uoh.edu.pk and Mukhtaj Khan; mukhtaj.khan@awkhum.edu.pk

Received 19 September 2021; Accepted 15 November 2021; Published 7 December 2021

Academic Editor: Naeem Jan

Copyright © 2021 Muhammad Fahad et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Euler path is one of the most interesting and widely discussed topics in graph theory. An Euler path (or Euler trail) is a path that visits every edge of a graph exactly once. Similarly, an Euler circuit (or Euler cycle) is an Euler trail that starts and ends on the same node of a graph. A graph having Euler path is called Euler graph. While tracing Euler graph, one may halt at arbitrary nodes while some of its edges left unvisited. In this paper, we have proposed some precautionary steps that should be considered in exploring a deadlock-free Euler path, i.e., without being halted at any node. Simulation results show that our proposed approach improves the process of exploring the Euler path in an undirected connected graph without interruption. Furthermore, our proposed algorithm is complete for all types of undirected Eulerian graphs. The paper concludes with the proofs of the correctness of proposed algorithm and its computation complexity.

1. Introduction

Graph is one of the discrete structures that consists of nodes (or nodes) and edges that connect these nodes. In order to avoid confusion, we will use the term node in the rest of paper. Generally, the graphs may be either directed or undirected [1]. Mathematically a graph can be defined as $G = (V, E)$ which consists of V, a nonempty set of nodes and E, and a set of edges [2, 3]. Each edge has either two nodes v and w, associated with it, called its endpoints. The edges are unordered pairs of nodes in undirected graphs, also represented as (v, w). In directed graphs, the edges are ordered pairs (v, w) of nodes, where v is the tail and w is the head of the edge. A path in a graph is a finite or infinite sequence of edges which connect a sequence of nodes which, by most definitions, are all distinct from one another [2, 3]. These paths can be used in designing the framework of a number of graph-related issues. For example, the graph model can be used in determining whether a communication link is established among different computers in a network or not. Furthermore, a number of issues such as planning efficient routes in mail and postage delivery, diagnosis of transportation system, etc., can be efficiently planned using paths in graphs. It is noteworthy that all graphs discussed in this paper are assumed to be undirected. Furthermore, it is also supposed that these graphs have a finite number of nodes and edges.

Among various types of paths in graph theory, Euler path is a special path that visits every edge of connected graph only once [4, 5]. In a connected undirected graph, there are no unreachable nodes and all the edges are bidirectional. A connected undirected graph will have an Euler path, but not an Euler circuit if and only if it has exactly two nodes of odd degree. The degree of node a is defined as the number of edges connected to that node. For the existence of the Euler path, it is necessary that exactly two nodes in a graph must have odd degree [4, 5]. A graph having the Euler path is called as the Euler graph. It is observed that while tracing the Euler graph, one may halt at any arbitrary node with some nodes (and its associated edges) remain unreachable.

RETRACTED

Some noteworthy classic work reported in [6, 7] concluded that the algorithm mentioned in Fleury's algorithm [8–10] is an elegant but inefficient algorithm, since, at the end of the algorithm, there are no edges left, and the sequence from which the edges were chosen forms of an Eulerian cycle if the graph has no vertices of odd degree. Similarly, in [10], the proposed algorithm may also be implemented with the Deque. Because it is only possible to get stuck when the Deque represents a closed tour, one should rotate the Deque by removing edges from the tail and adding them to the head until unstuck, and then continue until all edges are accounted for. These issues were addressed in our proposed approach.

In computers, there are a number of ways to represent graphs. One of these ways is to use adjacency lists. An adjacency list represents a graph as an array of linked list. The index of the array represents a node and each element in its linked list represents the other nodes that form an edge with the node [1, 2]. Figure 1 shows a simple graph G and its representation in adjacency list.

In this paper, we have presented an approach in exploring deadlock-free Euler paths in Euler graphs by revising adjacency list. The rest of this paper is organized as follows. Section 2 introduces the problem statement and traditional algorithm. In Section 3, the proposed approach is demonstrated in detail in comparison with conventional Euler path algorithm. Section 4 gives the proof of correctness of our proposed approach. Complexity of the proposed approach is given in Section 1. Some real-life applications of Euler path are discussed in Section 6. Concluding remarks are given in Section 7.

2. Problem Statement

Algorithm 1 depicts the construction of the Euler path. While tracing the Euler graph according to Algorithm 1, one may halt (or stall) at any arbitrary node and fail to visit the remaining unvisited edges of the graph thus leading to an incomplete traversal of the Euler path. This situation of halting may result in exponential increase of the computation cost even if a single node with its associated edges is added in the Euler graph.

Figure 2 shows Euler graph G and its two incomplete Euler paths halted at node a (Figure 2(b)) and b (Figure 2(c)), respectively.

The number on the edges shows the order of traverse. The order of traversal of the graph shown in Figure 2(b) is $b \longrightarrow a \longrightarrow c \longrightarrow d \longrightarrow a$ resulting to halting situation at node a, while the edges (c, e), (d, e), and (b, d) remain unvisited. Similarly, in Figure 2(c), the order of traverse is $a \longrightarrow c \longrightarrow b \longrightarrow d \longrightarrow a \longrightarrow b$ that also freeze at node a with (c, e), (c, d), and (d, e) left unvisited.

In other Euler graph H shown in Figure 3(a), two of many incomplete Euler paths halted at nodes d and b, respectively, are shown in Figures 3(b) and 3(c). The order of traversal of the graph of Figure 3(b) is $b \longrightarrow d \longrightarrow c \longrightarrow f \longrightarrow g \longrightarrow e \longrightarrow d$ and thus results in halting at node d. The same scenario exists in the faulty traversal shown in Figure 3(c).

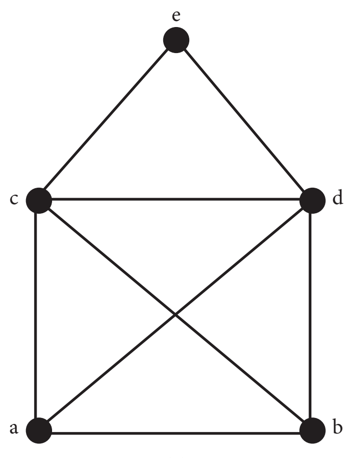
The *bridge* edge, as mentioned in Algorithm 1, is defined as an edge that when removed increases the number of connected components. The problem in faulty-Euler path lies when we accidentally visit the *bridge* edge. The procedure of finding the *bridge* edge by classical algorithm (Tarjan's bridge-finding algorithm) [13] is itself a complicated task for strong connected Euler graphs because it finds out the bridge edge(s) by extracting spanning forest from the graph. This process incurs space and time complexity $O(V \times (V + E))$ as computed by [1, 2]. Furthermore, the process becomes more complex when it tries to compute the size of multiplex computer network (represented by a graph) using conventional algorithm since, for every edge, another module is called in order to identify that the edge is *bridge* or not. Furthermore, the addition of extranode and its associated edges to an Euler graph (in a way that the resulting graph is also Euler) makes the graph too complex to explore the Euler path without being halted. In order to address this issue, we have proposed an informal solution to find the Euler path in any Euler graph without being halted, as discussed in Section 3.

3. Our Proposed Approach

Our proposed algorithm for finding deadlock-free Euler path in some Euler bidirectional connected graph is depicted in Algorithm 2. We applied our proposed algorithm on a number of Euler graphs. Figures 4 and 5 show the simulation results on Euler graphs of Figures 2 and 3. The results showed that our proposed algorithm outperformed the conventional approach. The first column of the table depicts the edge currently being visited. The edges to be visited are shown in column "Remaining edges." The last column shows the trace of our proposed approach to explore deadlock-free Euler path.

Still another classical example will clearly explain the performance of our proposed approach and help in exploring a deadlock-free Euler path. Consider a simple graph M (see Figure 6) having two odd-degree nodes c and d. If we explore the Euler path from any node other than c or d, we will be stuck with some edges remained unvisited. For example, the path $e \longrightarrow f \longrightarrow d \longrightarrow c \longrightarrow b \longrightarrow a \longrightarrow c$ will result to halt at node c with the edge (d, e) remain unvisited. If we apply our modified Euler algorithm, we can have deadlock-free path. Some of the many deadlock free Euler paths explored by our proposed algorithms are $d \longrightarrow e \longrightarrow f \longrightarrow d \longrightarrow c \longrightarrow a \longrightarrow b \longrightarrow c$ and $c \longrightarrow a \longrightarrow b \longrightarrow c \longrightarrow d \longrightarrow f \longrightarrow e \longrightarrow d$. It is noteworthy that both the paths start from one odd-degree node and ends on other odd-degree node.

Our proposed algorithm search for the odd-degree node first of an Euler graph from its revised adjacency list (see Figure 1) and then visit its adjacent even-degree nodes one by one prior to visiting the other odd-degree node. In other words, as long as there exist edges connected to even-degree nodes and traverse these edges prior to the other odd-degree node in order to avoid any deadlock until there is no choice.



node	Adjacent nodes
a	b, c, d
b	a, c, d
c	a, b, d, e
d	a, b, c, e

(a) (b)

Figure 1: A graph G (a) and its adjacency list (b).

```
**EularPathInConnectedGraph** ($G$: connected undirected graph with exactly two nodes of odd degree)
(1) Make sure the graph $G$ has 2 odd nodes.
(2) Start at any one of the two odd nodes.
(3) Traverse the edges one by one and then add in the array EP.
(4) While traversing a **bridge or** a **nonbridge** edge, select the nonbridge edge.
(5) Stop when you run out of edges.
    return EP
```

Algorithm 1: Euler path [11, 12].

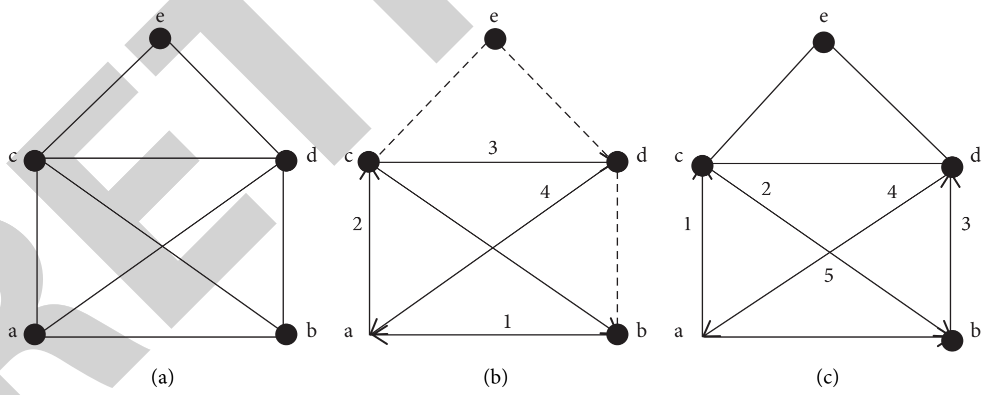


Figure 2: (a) Simple graph G. (b) A faulty-Euler path. (c) Other faulty-Euler path.

4. Proof of Correctness of Our Proposed Approach

Does our proposed approach always produce deadlock-free Euler path? The answer is yes. Let us prove by induction that each of the subgraph G_i, of a Euler graph G, where $i = 1, 2, \ldots, n$, that is generated by removing the visited edges one by one by our proposed approach (Algorithm 2), leads to a deadlock-free Euler path. It immediately concludes that the last subgraph G_n (having only one edge with two nodes) will have at least one odd-degree node. The induction phase is trivial since G_1 consists of a single edge connecting an odd-degree and even-degree node and also part of Euler path of the graph G. While solving the inductive step, we assume (for the sake of computation) that G_{i-1} is among one of the deadlock-free Euler paths. We need to prove that G_i, generated from G_{i-1} by our proposed approach, is leading towards a successful deadlock-free Euler path. We tried to prove this assertion by the contradictory rule by assuming

RETRACTED

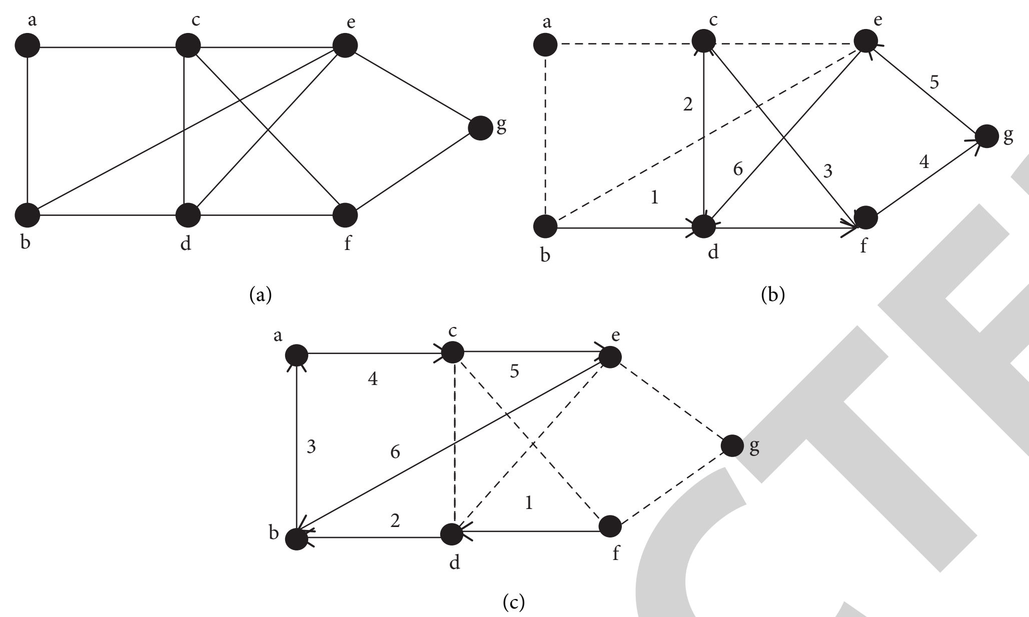
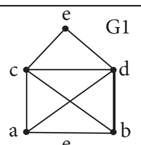
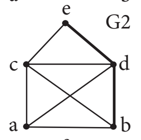
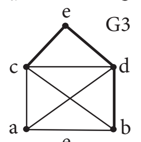
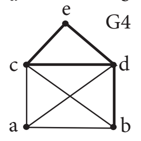
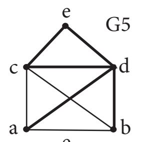
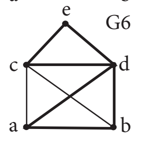
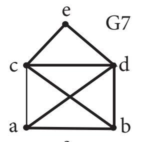
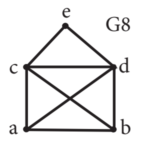


FIGURE 3: (a) Simple graph H (b) A faulty Euler path. (c) Other faulty Euler path.

Starting edge of graph	Remaining edges	Illustration
(b, d)	(d, e), (e, c), (a, c), (c, d), (a, d), (a, b), (b, c)	
(d, e)	(e, c), (c, a), (c, d), (a, d), (a, b), (c, b)	
(e, c)	(c, a), (c, d), (a, d), (a, b), (c, b)	
(c, d)	(c, a), (a, d), (a, b), (c, b)	
(a, d)	(a, c), (a, b), (b, c)	
(a, b)	(b, c), (a, c)	
(b, c)	(a, c)	
(a, c)	—-	

RETRACTED

FIGURE 4: Application of our proposed algorithm. The parenthesized labels of a node in the middle column indicate the nearest tree edge top to be selected; selected edges are shown in bold. The subscript in G_i depicts the order of edge visited.

RETRACTED

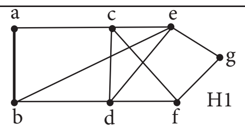
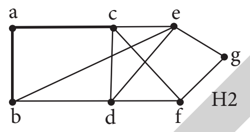
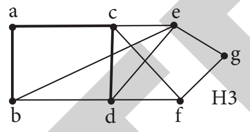
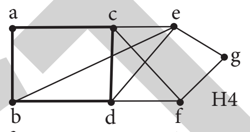
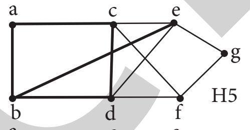
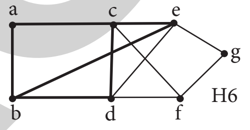
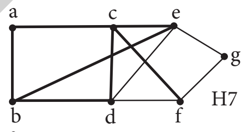
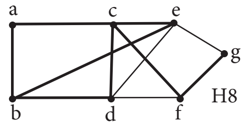
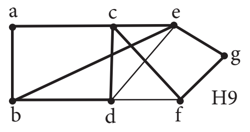
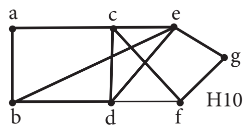
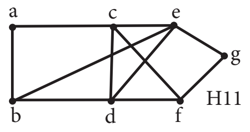
Starting edge of graph	Remaining edges	Illustration
(a, b)	(a, c), (c, e), (e, g), (f, g), (d, f), (b, d), (b, e), (c, d), (c, f), (d, e)	
(a, c)	(c, e), (e, g), (f, g), (d, f), (b, d), (b, e), (c, d), (c, f), (d, e)	
(c, d)	(c, e), (e, g), (f, g), (d, f), (b, d), (b, e), (c, f), (d, e)	
(b, d)	(c, e), (e, g), (f, g), (d, f), (b, e), (c, f), (d, e)	
(b, e)	(c, e), (e, g), (f, g), (d, f), (c, f), (d, e)	
(c, e)	(e, g), (f, g), (d, f), (c, f), (d, e)	
(c, f)	(e, g), (f, g), (d, f), (d, e)	
(f, g)	(e, g), (d, f), (d, e)	
(e, g)	(d, f), (d, e)	
(d, e)	(d, f)	
(d, f)	—	

Figure 5: Application of our proposed algorithm. The parenthesized labels of a node in the middle column indicate the nearest tree node to be selected; selected edges are shown in bold. The subscript in H_i depicts the order of edge visited.

that no Euler path of any undirected graph may have G_i. Let $e = (v, u)$ be the edge from a specific node of G_{i-1} to a node that is not in G_{i-1}. This assertion is used by our proposed method to expand G_{i-1} to G_i. This assertion initiated our proposed approach by the assumption that e cannot belong to deadlock-free Euler path including G. As a result, by adding e to G, a cyclic path is constituted (Figure 7). Furthermore, the edge $e = (v, u)$, in this cyclic path, must have the other edge (v', w') that is connecting a node v' of G_{i-1} to a node u' that is not in G_{i-1} (it is possible that $v\prime$ coincides with v or u' coincides with u but not both). By removing the edge $(v\prime, w\prime)$ from this cyclic path, the resultant graph is with two disconnected components since the only edge e is confirming the deadlock-free Euler path. Hence, this Euler path of G_{i-1} leads a deadlock-free Euler path which is clearly

RETRACTED

```
    **EularPathRevised** (G: undirected graph with exactly two nodes of odd degree)
(1) Make sure the graph G has 2-odd nodes.
(2) Start at any one of the two odd nodes v from the Degree column of new adjacency list.
    (a) Follow the edges to even-degree neighboring node w of node v.
    (b) Add the visited edge in the list EP and remove it from Graph G.
(3) If there is no choice then
    select the other odd-degree node
(4) else repeat step a and b.
(5) Stop when you run out of edges.
    return EP
```

Algorithm 2: Our proposed algorithm for Euler path.

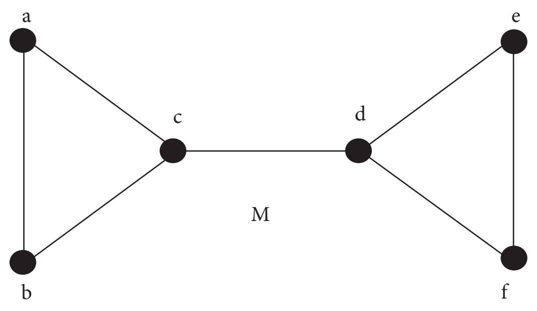


Figure 6: Classical Euler graph M.

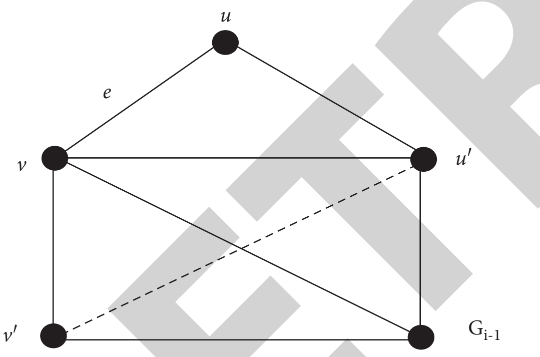


Figure 7: Correctness proof of our proposed approach (dashed edge is critical edge since its removal will produce the graph in two disconnected components).

contradicting the assumption that no deadlock-free Euler path will exist in graph G_i. This completes the correctness proof of our proposed approach.

5. Complexity of Our Proposed Approach

While computing the complexity of our proposed algorithm, we considered the revised adjacency list (see Figure 1). We start the Euler tour by selecting one of the odd-degree nodes from the revised adjacency list. This selection can be done in $O(1)$ time as the adjacency list is sorted in the descending order according to the degree of each node. After selecting one odd-degree node, we then select its neighboring even-degree node from the revised adjacency list that will take $O(e-1)$ because remaining edges are now $e-1$. This process will work in an iterative way until we have no choice other than visiting the other odd-degree node or all the edges are visited. It is pertinent to mention that the numbers of edges are equally important as the number of nodes since every edge has two end points and visiting one edge is the same as visiting the two nodes. Now, putting all together, we have the initial constant work done on first odd-degree node plus the work done on each edge on a Eulerian graph of n nodes, described above

$$2^{(n+1)/2}\pi^{-0.5}e^{\left(-n^2/2\right)+n(n-2)(n+1)/2}1+O\left(n^{-0.5+e}\right)\text{ for } n \longrightarrow \infty, \qquad (1)$$

where n is odd.

A detail discussion of the complexity of the proposed algorithm for Euler path is given in [14]. The proof of correctness of the complexity is given below.

5.1. Proof of Correctness.

This equation is derived from the classic work on the Euler path and circuits reported in [14, 15]. Since $n \longrightarrow \infty$, without loss of generality, we assume $e < 0.5$; then, the statement of the second part of equation (1) can be considered as random regular circular tournament on n nodes on average $2^{(n+1)}e^{0.5}n^{n-2}(1+O(n^{-0.5}+e)$ is considered or both the directed (and directed) spanning trees rooted at the nth node. It is noteworthy that it is $e^{0.5}$ more than the average for a random Euler tournament that needs not to be a regular tree.

It is noteworthy that these computations have several favorable interpretations in applied statistics models. Assume having an Eulerian graph G having start node v; the random walk may proceed to each adjacent edge that might be used at most once.

6. Applications of Euler Path

In real world, we come across many applications that ask for a path that traverses each street in a neighborhood, each road in a transportation network, or each link in a large computer network exactly once [1, 2, 12]. Among the other areas where Euler path are applied is in the layout of circuits, in network multicasting, and in molecular biology, where Euler paths are used in the sequencing of DNA [5].

7. Conclusion

In this paper, we have proposed an efficient approach in finding a deadlock free Euler path in the Euler graph. Our proposed algorithm presented here is in fact optimal to within a constant factor, since every edge and node of a graph must be examined in order to explore a deadlock-free Euler path.

Appendix

A. Complexity Analysis of the Proposed Euler Path

Let G be an Eulerian undirected having degrees $\deg_1$, $\deg_2$, ..., $\deg_n$ and let v be any arbitrary node, with deg as its degree; then, finding the probability P(G, v) that each edge of G will be encountered while tracing the Euler path, we can calculate

$$P(G, v) = 2^{m-d-1}\mathrm{Euler}(G)\sum_{j=1}^{n}\frac{(d_j/2)}{d_j}. \quad (A.1)$$

It is noteworthy that the probability that a particular Eulerian path can be traced by the random walk is inherently independent of the Euler path in each case.

For an undirected graph, the initial step of traversing the edge is 0.5. For the next step, the node v can be reached with the chance in deg2 of choosing the right edge out, then in deg4 the time after, and so on. Besides, for the nodes other than v, the first time for reaching is having only one chance in d_j1 of choosing the right edge out, one in d_j3 the time after, and so on, where d_j is the degree.

For an undirected graph, the primary step gets the opportunity 1/2 since in any edge the right way will help out for us in tracking down the "right way." For the subsequent time, node v is reached with the possibility in d_2 of selecting the right edge out, then at that node of degree with d_4, and so on. For the vertices other than v having degrees d_j1, d_j2, d_j3, and so on, there is an equally likely probability of selecting the correct edge to race the Euler path in a graph.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

All the authors contributed equally to this work.

References

[1] J. A. Bondy and U. S. Ramachandra Murty, *Graph Theory with Applications*, Macmillan London, London, UK, 1976.
[2] K. H. Rosen and K. Krithivasan, *Discrete Mathematics and its Applications: with Combinatorics and Graph Theory*, Tata McGraw-Hill Education, New York, NY, USA, 2012.
[3] D. B. West, *Introduction to Graph Theory*, Prentice Hall Upper Saddle River, Hoboken, NJ, USA, 2001.
[4] R. Thiele, "The mathematics and science of leonhard Euler (1707–1783)," in *Mathematics and the HistorianŠs*Springer, New York, NY, USA, 2005.
[5] P. A. Pevzner, H. Tang, and M. S. Waterman, "An eulerian path approach to DNA fragment assembly," *Proceedings of the National Academy of Sciences*, vol. 98, no. 17, pp. 9748–9753, 2001.
[6] G. Sánchez–Torrubia, C. Torres–Blanc, and L. Navascués-Galante, "EulerPathSolver: a new application for fleury's algorithm simulation," *New Trends in Intelligent Technologies*, vol. 14, pp. 111–117, 2009.
[7] A. Y. Saparov, "Recovering the recording sequence in scanned handwritten texts," *Vestnik Udmurtskogo Universiteta. Matematika. Mekhanika. Komp'yuternye Nauki*, vol. 28, no. 4, pp. 595–610, 2018.
[8] V. W. Guillemin and S. Sternberg, "An algebraic model of transitive differential geometry," *Bulletin of the American Mathematical Society*, vol. 70, no. 1, pp. 16–47, 1964.
[9] R. Fat, "A geometric construction situation with software assistance," *Research in Mathematics Education*, vol. 8, no. 3, pp. 195–230, 1987.
[10] H. Fleischner, *Eulerian Graphs and Related Topics Part 1, Volume 2, Volume 50 of Annals of Discrete Mathematics*, 1991.
[11] L. Euler, "Mechanica sive motus scientia analytice exposita: instar supplementi ad commentar," *Acad Scient Imper*, vol. 2, 1736.
[12] E. J. Barbeau and P. J. Leah, "Euler's 1760 paper on divergent series," *Historia Mathematica*, vol. 3, no. 2, pp. 141–160, 1976.
[13] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, 1972.
[14] W. T. Tutte and C. A. B. Smith, "On unicursal paths in a network of degree 4," *The American Mathematical Monthly*, vol. 48, no. 4, pp. 233–237, 1941.
[15] T. van Aardenne-Ehrenfest and N. G. de Bruijn, "Circuits and trees in oriented linear graphs," in *Classic Papers in Combinatorics*, pp. 149–163, Springer, New York, NY, USA, 2009.

RETRACTED