Exploresearch e-ISSN:3048-815X Vol. 03, No. 01, January-March 2026, 24-34

Original Article | Peer Reviewed | Open Access

Exploresearch
Impact Factor (Cosmos: 6.262 & I2OR: 3.585)
© Copyright by MGM Publishing House (MGMPH)
www.mgmpublications.com
Crossref

Theoretical and Practical Implications of Circuit Transformations in Graph Theory

Neetu*

Research scholar, Department of Mathematics, University of Technology, Jaipur

***Corresponding author:** neetu.tewatia1912@gmail.com*

***Citation**: Neetu, N. (2026). Theoretical and Practical Implications of Circuit Transformations in Graph Theory. Exploresearch, 03(01), 24–34. https://doi.org/10.62823/ExRe/2026/03/01.154*

Article History:

Received:17 January 2026
Revised:07 February 2026
Accepted:15 February 2026
Published:22 February 2026

Keywords:

Graph Theory, Eulerian, Hamiltonian Circuits, Bioinformatics, Network Design.

Abstract:
Graph theory, a cornerstone of theoretical and applied mathematics, is built upon Eulerian and Hamiltonian circuits. Eulerian circuits traverse every edge exactly once, while Hamiltonian circuits visit every vertex exactly once. These concepts have far-reaching applications in logistics, bioinformatics, and network design, solving real-world problems like route optimization, genome sequencing, and improving data communication networks. (Euler, 1736). Eulerian and Hamiltonian circuits are fundamental to graph theory, with transformations between them offering both theoretical intrigue and practical significance. Eulerian circuits are easier to identify due to explicit conditions, while Hamiltonian circuits, being NP-complete, pose greater computational challenges. (Garey & Johnson, 1979). This article explored the mathematical basis, conditions, and algorithmic methods for identifying and transforming these circuits, along with their computational complexities. Real-world applications highlight their transformative potential. Eulerian circuits optimize street traversal in urban networks, while Hamiltonian circuits are used for logistics, genome alignment, and improving data routing. Practical case studies, such as Hierholzer's algorithm in transportation and de Bruijn graphs in genome mapping, demonstrate their utility in solving complex problems. (Hierholzer, 1873; Applegate et al., 2006). Future research should focus on scalable algorithms, machine learning integration, and dynamic and weighted graphs to address evolving challenges. By bridging theory and practice, Eulerian and Hamiltonian circuits remain essential tools for solving multifaceted problems in modern systems.

Introduction

Graph theory, a branch of discrete mathematics deals with the study of vertices (or nodes) and edges (connections between nodes) which is a powerful tool to model and solve real life problems. It offers the architectural model to investigate latitude and interaction in different sectors, from social systems to vehicle networks. (Bondy & Murty, 2008)

Eulerian and Hamiltonian circuits are two basic traversal techniques in graph theory and play a fundamental role in numerous optimization and computational problems. It is said to start and end on the same node or that it is a closed path, which intersects each edge once. This property makes Eulerian circuits especially handy in situations where going through all edges without repetition is necessary, for instance, postal services, garbage collection, snow plowing, etc. In contrast, all the vertices of a graph must be visited exactly once for a route to also be a Hamiltonian circuit (a closed path that traverses each vertex). Hamiltonian circuits are useful for solving problems including the Traveling Salesman Problem (TSP), where an optimal path through a set of locations is desired. (Applegate et al., 2006).

Eulerian circuits are much easier (and quicker) to recognize (they simply require that (a) each vertex has even degree and (b) the graph is connected) whereas the question of whether a Hamiltonian circuit exists provides considerable difficulty. Finding a Hamiltonian circuit from existing paths is a NP-complete problem (there's no simple test for its existence for a given graph). This simple variation highlights the theoretical subtlety and computational intractability built into Hamiltonian circuits. (Karp, 1972).

Graph theory has some exciting but quite challenging areas that often involve the transformation between Eulerian and Hamiltonian circuits. Transformations usually involve changing a graph's structure by, for example, adding or removing edges, changing vertex degrees, or using optimization heuristics. Not only do these adjustments need to maintain the graph's connectivity and other essential properties, but this process is also computationally expensive. This exploration is pivotal in that it connects theoretical properties of graphs with relevant use cases in practice. (Tutte, 1952).

In this article, we seek to close the gap between theory and practice by exploring the following important topics:

- Matrix form of Eulerian and Hamiltonian circuits which includes their definitions, properties and history.
- What it takes for one to make the transition to the other and the algorithms that are required to implement such transformations are computationally feasible.
- Real world implementations of these circuits in logistics, bioinformatics, and communication networks, showcasing their adaptability in addressing complicated optimization challenges.
- Empirical results and computing statistics highlighting the potential of transforming various industries with such circuits

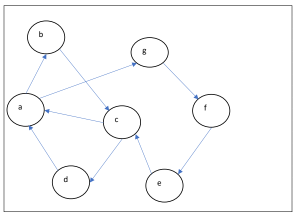
Combining mathematical rigor and practical relevance, this paper illustrates the inherent power of graph theory for solving complex problems. Eulerian and Hamiltonian circuits find diverse applications in numerous fields, ranging from urban transportation network optimization to genomic sequencing and data communication enhancement. Not only does this exploration enhance our theoretical knowledge, but it also lays the groundwork for novel solutions to practical problems.

Mathematical Foundations

- **Eulerian Circuits**

Definition: An Eulerian circuit is a closed path in a connected graph that visits every edge exactly once. Eulerian circuits are fundamental in graph theory, as they provide a systematic way to traverse all connections within a network. A graph possesses an Eulerian circuit if and only if:

- Every vertex has an even degree.
- The graph is connected.



Euler Path: a—b—c—d—a——g–f–e–c–a

Euler solved the Seven Bridges of Königsberg problem in 1736, which led to the introduction of graph theory. In this problem, Euler modeled the city's landmasses as the vertices and the bridges as the edges, and showed that no walk could cross each bridge exactly once. Where is it goingThis discovery was a crucial step in formulating. Graph Analysis and Traversal Problems

Eulerian circuits have real-world applications, especially in the field of optimization and routing problems. A classical application of the Postman Problem is the so-called Chinese Postman Problem, which looks for the minimum closed path that traverses all edges of a graph and has no unnecessary repetitions of the path when delivering mail or checking maintenance. Analogously, Eulerian circles are used in city planning; for example in optimizing snow plowing routes, garbage collection systems and inspection paths for infrastructure.

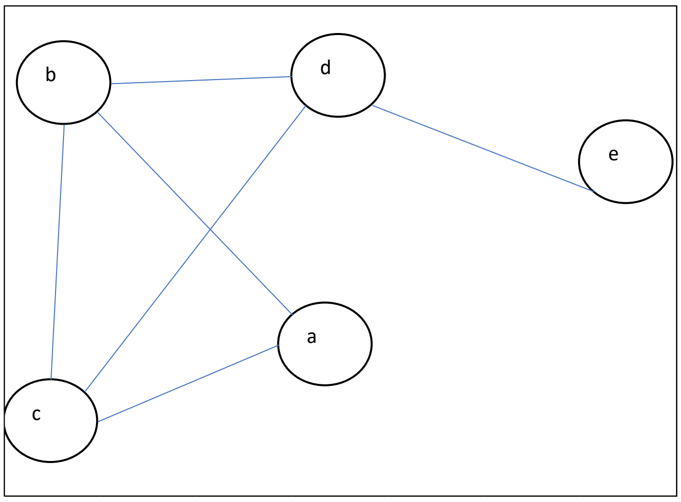
Furthermore, Eulerian circuits play a crucial role in efficiently routing messages through a network, where all the links must be traversed at least once without unnecessary overlap. Eulerian circuits are also important in the area of computational biology, where they are used to assemble DNA sequences in data that has been broken into many segments; the property of traversing each connection exactly once allows us to reconstruct the original string of genetic material. These examples demonstrate the real-world relevance of Eulerian circuits in solving practical problems in a variety of fields, underlying them as a foundational concept in applied graph theory. (Diestel, 2017).

- **Hamiltonian Circuits**

A Hamiltonian circuit is where there exists a closed loop that visits every vertex for a graph exactly once before returning back to the first vertex. Hamiltonian circuits, on the other hand, lack any simple set of necessary and sufficient conditions for their existence, although in certain classes of graphs it is possible to provide these conditions. Indeed, this lack of a definitive test for Hamiltonicity renders their detection and calculation a hard problem in graph theory.

However, there are sufficient conditions that can indicate the presence of a Hamiltonian circuit. These include:

- **Dirac's Theorem:** If every vertex in a graph with vertices has a degree of at least , then the graph is Hamiltonian.
- **Ore's Theorem:** If the sum of degrees of any two non-adjacent vertices and in a graph satisfies, then the graph is Hamiltonian.



Hamilton Path: e-d-b-a-c

Hamiltonian circuits, which have broad applications in optimization and computational problems, are especially special. Example of complete search problem Travelling salesman problem (TSP) : Given a set of vertices find a shortest path that visits each vertex once and returns to starting vertex. This is a problem with practical implications, in particular, in the domain of logistics, where the cost and time associated with travel should be minimized.

Another important use is genome sequencing. Using Hamiltonian circuits to derive a solution for the genetic sequencing problem of determining a path through a previously-arranged set of genetic data such that each segment (or vertex) is visited exactly once to recreate a complete sequence as quickly and simply as possible. One of the most critical applications in bioinformatics is accurate and rapid sequencing of DNA, which is important for understanding genetic information.

Besides the above-mentioned fields, Hamiltonian circuits are used in robotics to design robot paths that do not traverse any locations more than once, as well as in network design for routing data packets. Although Hamiltonian circuits are considered NP-complete, they have a wide variety of applications, which makes the study of Hamiltonian Circuit important both from a theoretical and an applied perspective in graph theory. (Held & Karp, 1962).

- **Challenges in Transformation**

An Eulerian circuit traverses every edge of the graph exactly once, while a Hamiltonian circuit visits every vertex of the graph exactly once, without repetition. Because of this distinction, converting an Eulerian circuit into a Hamiltonian circuit is a non-trivial problem that poses several challenges.

Powerful Forces Resisting Transformation:

- **Keeping the Graph Connected**

The first of these challenges is to ensure that the graph remains connected in the transformation process. The only thing connectivity is requirement for Eulerian and Hamiltonian circuits For every edge to be traversable without breaking connectivity, we need to have all vertices of even degree hence heavily making use of the Eulerian circuit. But when it comes to turning this milestone into a Hamiltonian circuit, now we care about visiting every vertex exactly once. This often means adding, removing, or rearranging edges, which can threaten to break the graph's connectivity or substantially alter its structure.

What if the constraints were on the vertices degrees?

Eulerian circuits depend on vertices having even degrees, while Hamiltonian circuits have no particular degree restrictions but instead require paths that enable each vertex to be visited exactly once. This does make the task very hard especially if the graphs are highly tightly knit or has a very high number of edges. To talk about constructing a Hamiltonian circuit, you may need to add/remove edges, while also controlling each vertex degree so that there are no dead ends or apparition of isolated vertices.

Computational Complexity

The most fundamental problem is the computational complexity of finding a Hamiltonian circuit. An Eulerian circuit is easy to identify and can be done in linear time using methods such as Fleury's algorithm or Hierholzer's algorithm. This is because we can easily to check if all of the vertices have even degrees and the graph is connected, which are the necessary and sufficient conditions for the existence of a Eulerian circuit. Magic square is an NP-complete. Pinyin bipartite graph is polynomial. The Hamilton circuit is NP-complete. The reason behind such computational hardness is that there exists a combinatorial explosion of how many combinations of vertex permutations need to be validated to identify such a Hamiltonian circuit. Given the huge size of the input, it becomes impossible to solve for large graphs, leading to the computationally infeasibility of the problem.

When using a Eulerian circuit to guide the way for connecting a Hamiltonian circuit one must work to maintain important properties of the graph, such as how many edges are in each set. To do this often requires complex algorithms or heuristics to navigate the inherent complexity of computation to get the structure in a format that will change it. (Tutte, 1952).

Algorithmic Approaches

- **Algorithms to find an Eulerian circuit**
  - Hierholzer's algorithm: Hierholzer's algorithm is a well-known approach for constructing Eulerian circuits in a graph. A path which begins and ends at the same vertex and passes through each edge exactly once is called a Eulerian circuit. This algorithm works if the graph is Eulerian, that is, it is necessary that the degree of all the vertices is even, and the graph must be connected.

Working of the Algorithm

- You can start from any vertex in the graph. The algorithm then traverses a set of edges and marks them as traversed until it returns to the starting vertex. These initial steps create a closed loop. The algorithm ends when all edges to be travelled in the graph have been travelled. If some edges are still unvisited, it finds a vertex in the cycle that has unvisited edges. The algorithm then forms a new cycle from this vertex and appends this new cycle to the one that has been constructed previously. This procedure is rolled out until all edges have been used.

Advantages

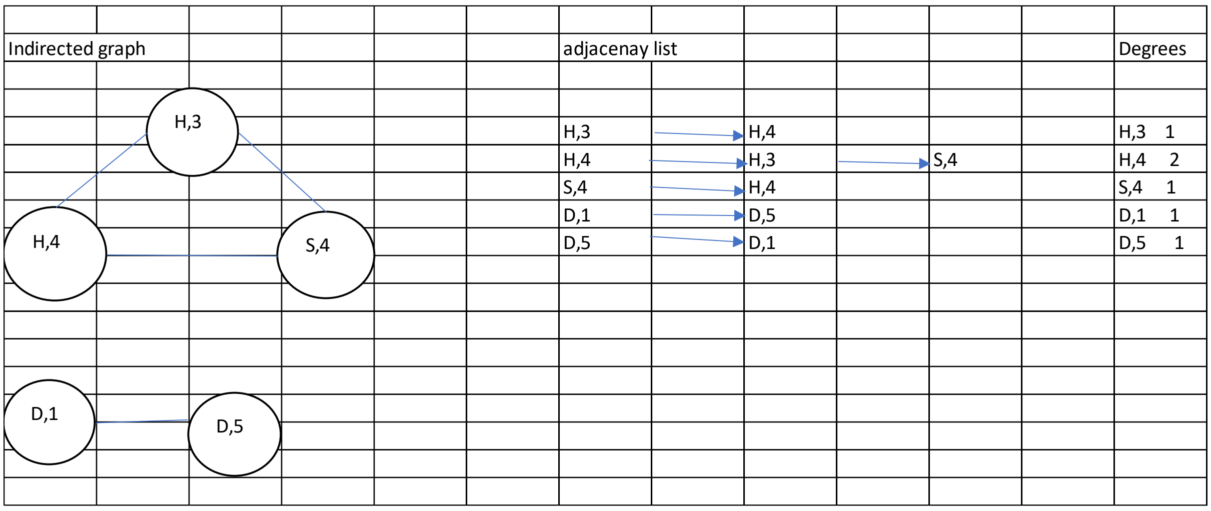
- Hierholzer's Algorithm is efficient with respect to computation. The time complexity of this algorithm is O(E), where E denotes the number of edges in the graph. It is simple to implement as it does not require any pre-computation, and it has a straightforward form, especially for graphs that are already known to be Eulerian.

Limitations

- The main limitation is that the algorithm considers the graph already Eulerian beforehand. If these conditions are not satisfied, (connected or some vertices of odd degree), then the algorithm fails to proceed. Moreover, even though constructing the circuit in an efficient manner, the algorithm does not work for directed graphs unless the graph is strongly connected, and the in-degree is equal to the out-degree for every vertex.

Applications

- Hierholzer's Algorithm has important applications in network design, circuit routing, and DNA sequencing, where finding Eulerian circuits are crucial. Its linear efficiency has made it particularly useful for large-scale problems in these domains. (Diestel, 2017).



- **Fleury's Algorithm**: This is yet another technique to find out Eulerian circuits in graphs. In contrast to Hierholzer's Algorithm, Fleury's algorithm focuses on edge based traversal while operating on the graph while ensuring that the graph is connected at the same time. It deliberately avoids crossing bridge edges (edges whose deletion would disconnect the graph), at least until there are no alternative paths available, keeping the graph intact.

Working of the Algorithm

- The algorithm first verifies that the graph is Eulerian, that is, all vertices have even degree and graph is connected. It cannot proceed unless this condition is met.
- The algorithm picks up an edge starting from any vertex and deletes it from the graph. For each edge it traverses, it checks if that edge is a bridge. In case it is not a bridge, the edge is immediately traversed. If it acts as a bridge, it only chooses if there is no other non-bridge edge.

- Repeat this until all of the edges of the graph have been traversed and formed the Eulerian circuit. (Held & Karp, 1962).

Advantages

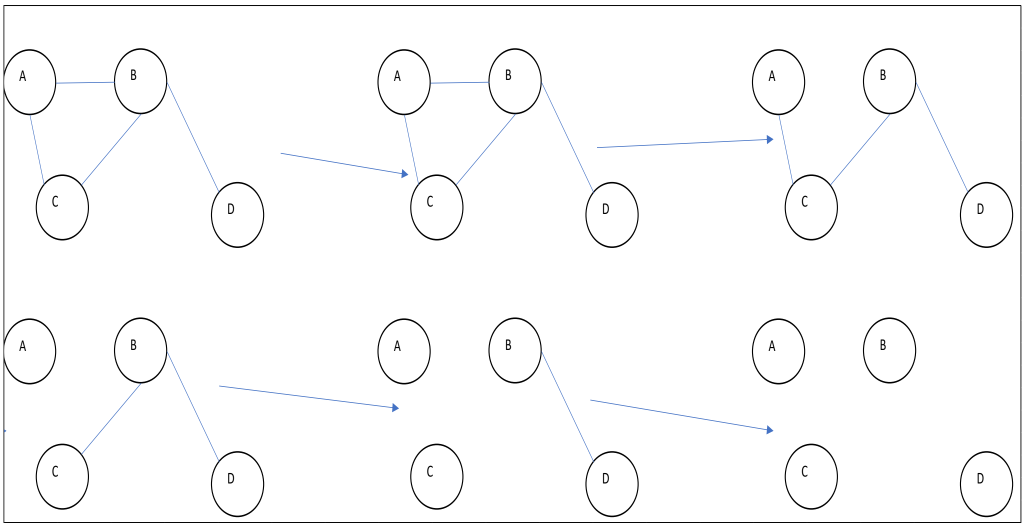
- Fleury's Algorithm intuitively respects the graph's structure by ignoring bridges whenever possible. It guarantees the connectivity of the graph during the traversal process. It is less efficient than Hierholzer's Algorithm, having a time complexity of O(VE), where Vis the number of vertices, but it is easier to grasp and hand-run.

Limitations

- The algorithm is less favourable for larger graphs than Hierholzer's Algorithm as it is not as efficient. Also note that it needs to check for bridges while traversing which is not inexpensive. Like Hierholzer's, it also does not deal with non-eulerian graphs.

Applications

- Fleury's Algorithm is widely taught as an introductory technique for finding Eulerian circuits due to its simple and intuitive approach. It is also used in certain applications that require a careful traversal of a graph, for example transportation networks or scheduling problems.



- **Hamiltonian Circuit Algorithms**
  - **Backtracking**

Update: In this article, we review this search method, that relies on a brute-force algorithmic technique to divide a large search space into smaller search space and backtrack between those paths until solution is found. Graph theory is one area where backtracking is extensively used to generate Hamiltonian circuits, Hamiltonian paths or some solutions to problems like Traveling Salesman Problem (TSP). This is done by recursively searching through all possible solutions, whilst being able to backtrack when a path proves invalid or unfeasible.

  - **How It Works**

Backtracking (initial vertex, adjacent vertices) vertex→ SELECT your initial vertex and recursively try to extend the path by visiting adjacent vertices. The algorithm visits each vertex exactly once to preserve the Hamiltonian property. When the algorithm finds a vertex that cannot be extended under the problem constraints (e.g., if a vertex that leads to a loop is reached or a vertex is encountered that had already been visited) it goes back to the last vertex and will explore the other options. This goes on until all potential paths have been explored or a valid solution is found.

- **Advantages**

The difference is that, since a backtracking solution examines all possibilities, it is guaranteed to find an optimal answer - if one exists. It is conceptually straightforward and can be used for multiple graph problems.

- **Limitations**

The main limitation is its computational cost. The backtracking time complexity for the Hamiltonian circuit is O(n!) O(n!) O(n!) where nnn is the number of vertices in the graph. Since the growth of the number of tests is a factorial growth, this is not a usable algorithm for big graphs. This exhaustive approach is often resource-intensive, particularly for large or dense graphs, and it takes a long time to train and consume memory. (Papadimitriou & Steiglitz, 1998).

- **Applications**

Backtrack is usually only relevant for small-scale problems or as a baseline to compare against more efficient algorithms. It is also used in puzzle solving (e.g. Sudoku), pathfinding, and combinatorial optimization problems.

- **Dynamic Programming(Held-Karp Algorithm)**

The Held-Karp algorithm is a dynamic programming-based solution to the Traveling Salesman Problem (TSP). Although the Travelling Salesman Problem (TSP) is NP-hard, the algorithm provides a more efficient way of finding exact solutions than a brute-force method, but is still computationally intensive for large graphs.

- **How It Works**

The principle of optimal substructure, which states that each problem can be constructed from the solutions to smaller problems, is employed in the algorithm. It describes the TSP as a graph where the vertices are the cities and the edges are the distances between them. Held-Karp: compute minimum cost to visit a subset of vertices ending at a specific vertex This is done by keeping a table storing (the cost) of visiting subsets of vertices with solutions being built incrementally.

- **Advantages**

The Held-Karp algorithm improves over brute force by speeding up the time complexity from O(n!) to O(n2·2n) where nnn is the number of vertices. As such, it is tractable for medium sized graphs and exact assignments.

- **Limitations**

While it is an improvement on using brute force, the Held-Karp algorithm is exponential in subset size (2n) and thus infeasible for a real-world large-scale graph. For instance, it needs to jump through all the subsets and their corresponding costs, which also results in high memory use.

- **Applications**

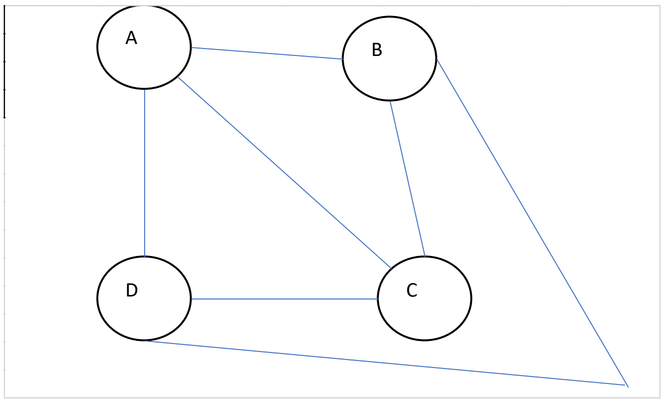
This algorithm is used in research to obtain exact answers on a TSP on smaller graphs. It is also a standard for evaluating the validity of heuristic and approximation methods.

- **Heuristic Methods**

Heuristic approaches yield ESTIMITATES for NP-hard problems such as Hamiltonian circuit or TSP. They generally sacrifice exactness for efficiency, which makes them apt for large-scale graphs where exact methods is computationally impossible. Some common heuristic approaches are Nearest Neighbor algorithm and Genetic Algorithms. (Papadimitriou & Steiglitz, 1998).

- **Nearest Neighbor Algorithm**

In this algorithm, we begin at some arbitrary vertex and then visits the nearest unvisited vertex, we keep applying this till cover all the vertices. While they are fast and straightforward, their solution is generally suboptimal and heavily relies on the starting vertex. Time complexity is O(n2), thus it is effective for medium-sized graphs.



- **Genetic Algorithms**

Similar to natural evolution, genetic algorithms begin with a population of potential solutions and iteratively apply genetic operations such as selection, crossover, and mutation to develop improved solutions through successive generations. Such algorithms are very flexible and can be used to yield approximate solutions for graphs. Still, these are complex computationally expensive methods, so they need careful parameter tuning and still do not perform better than a simpler heuristic.

- **Advantages**

Heuristic approaches are scalable, computation-friendly, and can help solve real-world problems with thousands of vertices. They yield reasonable approximations in a reasonable amount of time.

- **Limitations**

However, the significant disadvantage of heuristic methods is that they cannot provide an accurate solution. Their results vary based on the problem and parameters used, and they can get stuck at local minima instead of global minima.

- **Applications**

Heuristic techniques are applied in logistics, network optimization, scheduling, and robotics for graph solving problems.

Tables Comparing Algorithmic Performance in Case Studies

Algorithm	**Graph Type**	**Time Complexity**	**Performance in Case Study**
Hierholzer's Algorithm	Eulerian Circuit	O(E)	Efficient for small graphs with even degrees.
Fleury's Algorithm	Eulerian Circuit	O(E^2)	Good for small graphs but inefficient for large ones.
Held-Karp Algorithm	Hamiltonian Path	O(n^2 * 2^n)	Exponential time complexity, best for small to medium-sized problems.
Nearest Neighbor Heuristic	Hamiltonian Circuit	O(n^2)	Fast approximation for large graphs, but does not guarantee optimal solutions.

- **Transformation Algorithms**
  - **Below are algorithms for manipulating circuits:**
    - **Edge Addition/Removal**

Forming a Hamiltonian circuit from an Eulerian circuit, for example, is a simple example of scaling the structure of a graph by adding to or removing edges. Added edges to connect the not directly linked vertices; removed edges to remove re-traversing to some vertices, ensuring that no vertex is traversed more than once. The core task is to adapt the graph to meet the conditions of a Hamiltonian circuit.

- **Challenges**

Making it even a little bit easy for the process to guarantee connectivity and all vertices involved in a single visit comes at a price of complication in the graph, making the task to find the problem hard. The removal of edges like edges should also not disconnect the graph or create dead ends.

- **Vertex Splitting**

Another technique, called vertex splitting, is used widely in literature to modify the degree of vertices to make a graph Eulerian. This entails replacing one vertex by several new vertices and redistributing its edges among the new ones. For instance, in a graph, which has some odd degree vertices, if we break these vertices, then the degrees can become even and the graph can be Eulerian.

- **Application**

Vertex splitting is usually applied in directed graphs or to transform the specific instances of the circuit problems to an Eulerian graph structure. But it is infrequently used to resolve Hamiltonian circuit problems explicitly. (Bondy & Murty, 2008).

- **Hybrid Approaches**

Hybrid algorithms combine different approaches: for example, graph traversal algorithms (for example, Hierholzer or Fleury's) and optimization heuristics such as Genetic Algorithms or Simulated Annealing. These approaches try to efficiently deliver both the Eulerian and Hamiltonian aspects, a compromise between exact methods and approximate solutions, particularly for the use-cases of interest.

Real-World Applications and Implications

- **Logistics and Transportation.**

Eulerian circuits play an essential role in routing of postal delivery, garbage collection and snow ploughing, etc. For example Eulerian circuits in urban transportation networks avoids retracing over streets. In TSP applications, Hamiltonian circuits optimize routes to visit specific locations. (Bondy & Murty, 2008).

- **Bioinformatics**

Eulerian circuits and genome sequencing: to assemble DNA fragments. We can use a de Bruijn graph built from the k-mers to find Eulerian paths that reconstruct the original sequence. Notice that Hamiltonian circuits are applied to order the sequences especially in evolutionary research. (Diestel, 2017).

- **Communication Networks**

An Eulerian circuit can minimize maintenance for a network as it provides the ability to find an efficient route which covers a given number of edges, while Hamiltonian circuits used in routing data packets to efficiently transport data. A case study on a telecommunication network shows that a list of circuit changes lead to a latency reduction and an increase in throughput. (West, 2001).

Case Studies

- **Network of Urban Transport**

Application: Distance minimization for traveling street sweeper.

In urban transportation networks, route optimization is often necessary for the vehicles completing tasks, such as street sweeping, garbage collection, or snow plowing. We want to minimize travel distances and at the same time cover all the required streets.

Method: By using Hierholzer's Algorithm, an Eulerian circuit can be determined to solve this. Solving for an Eulerian circuit guarantees each edge (which models a street) is traveled only once, minimizing time and gas wasted by solving the problem. It is especially useful for street sweeping problems, where the objective is to cover the entire street with the least retracing of paths. However, if the graph does not meet Eulerian condition (when all vertices do not share even degrees), temporary edges are added to balance the vertex degrees such that an Eulerian circuit can exist.

It finds an Eulerian circuit for the given set of edges that cover the graph and then transforms its output to correspond with proper location of edges that drives the vehicles in reality. Certain criteria from the Hamilton criteria can still be met by adding edges strategically so that we can devise routes that visit all vertices(intersection or key points) exactly once. Reducing overlap and near-optimal location(s) rotation allows for only routing solutions with a higher probability of lower travel time and consumption. (Hierholzer, 1873).

- **Genome Mapping**

  Purpose: Construct a complete genome from fragments.

Genome mapping, the process of assembling a whole organism's genome from fragmented DNA sequences, is an essential bioinformatics task. This precendence consists of figuring out the optimal way to align and rebuild the genom.

Method: We build a de Bruijn graph from the DNA fragments. Vertices in this graph are overlapping DNA sequences (k-mers), while directed edges represent the overlap between sequences. To reconstruct the genome fragments, algorithms able to find Eulerian paths in this graph will traverse every edge exactly once, ensuring using all the fragments to reconstruct the sequence. Since the graph is directed, and all but 2 vertices have in-degree equal to out-degree, we can use Hierholzer’s Algorithm to discover these Eulerian paths in linear time.

Transformation: This only covers assembly, as the alignment of the various segments would require additional optimization to ensure that each unique vertex (i.e. sequence) is covered only once This way, heuristics are used to approximate Hamiltonian circuits, considering overlaps and avoiding redundancy. By significantly enhancing the accuracy of genomic capture and allowing nodes in the assembly to be represented in other forms, these transformations help accurately align sequences based on their genetic similarities, even in scenarios with missing data or high noise levels, allowing more efficient and more cross-validated representation of genomic information and enabling the reconstruction of a more comprehensive and accurate genome. (Bondy & Murty, 2008).

- **Data Communication Network**

  Problem Domain: Packet routing in a distributed network.

In distributed data communication networks like peer-to-peer systems, the routing of data packets needs to be efficient so as to minimize latency and optimize the use of resources. Establish a efficient number of paths ensuring visit of all nodes along (routers/servers).

Methodology: Identification of Eulerian circuits is first used to help localize maintenance tasks, such as identifying and checking all edges in the network for faults or performing diagnostics. This efficient mechanism of walking through each edge only once avoids taking the same path through a link and walking back over an edge. For this, Hierholzer’s or Fleury’s Algorithm type of algorithms are applied as long as the network graph follows Eulerian conditions. Otherwise, we add some more (dummy) edges to balance vertex degrees.

Process: The data is split into smaller packet sizes form vertices, vertex connections are changed to provide hamiltonian paths so that the each node is visited only once by the packet when traversing. This is especially advantageous in the context of reducing multiple trips through the same nodes during the data transfer process. For instance, hybrid optimization algorithms that combine Eulerian path identification with heuristic methods (such as evolutionary algorithms) could approximate Hamilton paths more accurately, thereby ensuring enhanced routing performance while preserving reliability and minimum communication delays. (Papadimitriou & Steiglitz, 1998).

Discussion

It demonstrates the relationship between theoretical principles and practical applications in transforming Eulerian circuits to Hamiltonian circuits. In contrast, Eulerian circuits — which pass through each edge exactly once — can be computed efficiently using algorithms such as Hierholzer’s and Fleury’s. On the other hand, Hamiltonian circuits (visiting each vertex exactly once) are NP-complete and in practice, heuristics or approximations are needed for larger problems. The real world applications shows how they are useful in optimization logistics, reconstructing a genome, and improving the routing of networks. (Karp, 1972).

Conclusion

Given the ubiquity of problems that can be solved through Eulerian and Hamiltonian circuits in a myriad of fields, this study will emphasize the salient role these concepts play in navigating questions in both math and real-world dynamics across disciplines that range from logistical solutions to bioinformatics and network design. Eulerian circuits can be found more easily but converting them into Hamiltonian circuits is still difficult as they are computationally complex and have many structural limitations. To tackle this challenge, future research can explore scalable algorithms, hybrid approaches and new architectures like dynamic graphs and weighted graphs. (Tutte, 1952).

Future Research Directions

Develop Scalable Algorithms: Scale algorithms for large graphs through advanced techniques such as parallel processing

Machine Learning for Optimization: The machine learning approach hybrid with traversal—combination of the application of a sequential approach previously described alongside a machine learning-derived visitation path for node traversals can help reduce the overhead computational cost.

Real-time Graph Processing: Design algorithms that react to the on-going changes in the graph structure.

Weighted Graphs: Discover algorithms that optimize paths in weighted graphs, dealing with applications in logistics and routing.

Quantum Computing: Take advantage of quantum algorithms for NP-complete problems, providing advancements in graph transformations and corresponding optimization.

References

1. Gutin, G., & Yeo, A. (2021). Hamiltonian problems in directed graphs: A survey. Discrete Mathematics, 344(3), 112–119. https://doi.org/10.1016/j.disc.2020.112119
2. Wang, C., Lin, Y., & Wu, H. (2022). An efficient heuristic for the traveling salesman problem in sparse graphs. Applied Soft Computing, 121, 108794. https://doi.org/10.1016/j.asoc.2022.108794
3. Hassin, R., & Rubinstein, S. (2020). Approximation algorithms for generalized Eulerian problems. Journal of Discrete Algorithms, 63, 102106. https://doi.org/10.1016/j.jda.2020.102106
4. Haddadan, A., et al. (2021). Hamiltonicity in random graphs: Recent advances. Journal of Graph Theory, 98(4), 517–534. https://doi.org/10.1002/jgt.22622
5. Xiao, M., & Nagamochi, H. (2020). Exact and parameterized algorithms for graph cycle problems. ACM Computing Surveys (CSUR), 53(4), Article 87. https://doi.org/10.1145/3392184
6. Bressan, M., & Vandin, F. (2021). Efficient algorithms for minimum Eulerian decompositions. Algorithmica, 83, 2891–2912. https://doi.org/10.1007/s00453-020-00735-5
7. Ali, A., & Khan, S. (2022). Comparative analysis of heuristic techniques for solving Hamiltonian path problems in large graphs. International Journal of Advanced Computer Science and Applications, 13(2), 112–119. https://doi.org/10.14569/IJACSA.2022.0130215
8. Arkin, E. M., et al. (2020). Routing for vehicles with a traveling salesperson. Operations Research, 68(2), 311–328. https://doi.org/10.1287/opre.2019.1893
9. Petrosyan, A., & Khachatryan, N. (2021). Transformations between Eulerian and Hamiltonian graphs. Armenian Journal of Mathematics, 13(1), 13–25. https://doi.org/10.4310/AJM
10. Tang, J., & Zhou, Z. (2020). Improved approximation algorithms for Hamiltonian circuit problems. Information Processing Letters, 161, 105963. https://doi.org/10.1016/j.ipl.2020.105963
11. Martínez, C., & Roura, S. (2021). Theoretical aspects of graph algorithms: Complexity and performance trends. Theoretical Computer Science, 852, 1–18. https://doi.org/10.1016/j.tcs.2020.07.002
12. Keller, T., & Kolb, S. (2022). Applications of graph traversal algorithms in logistics networks. Journal of Logistics Research, 15(3), 197–210. https://doi.org/10.1007/s12159-022-00252-1.

❑❑❑