Review Article

Comparing Traversal Strategies: Depth-First Search vs. Breadth-First Search in Complex Networks

Abstract

This article compares and contrasts two basic graph traversal algorithms that are commonly employed in computational problem-solving and network research. Common applications of these algorithms include pathfinding, optimisation of network flows, collaborative exploration, and classification tasks. To find out how well they function with different types of datasets, network topologies, and issue domains, researchers have systematically reviewed previous works. We measured the efficiency of each solution using performance indicators like execution time, memory utilisation, and path length. According to the results, one approach is more effective in memory-constrained settings and deep searches, while the other is better at discovering the shortest paths and providing comprehensive coverage. Furthermore, the paper emphasises the advantages of hybrid techniques, which merge the best features of both algorithms to provide better results in specific cases. This comparison helps fill gaps in our knowledge of graph-based problem-solving methods and sheds light on how to choose the best traversal algorithms for different types of applications.

.

Keywords: Graph Traversal Algorithms , Depth-First Search (DFS) , Breadth-First Search (BFS) , Complex Network Analysis , Algorithm Performance Comparison

1. Introduction

Numerous industries, including the tourist industry, have seen profound shifts as a result of the expansion of information technology. Providing visitors with opportunities to learn about and engage with a destination's rich history and culture is the essence of historical and cultural tourism.[1]. It has emerged as the top alternative travel destination for tourists who want to discover historical sites and cultures with historical significance. To enhance the experience of the tourists, it is crucial to maximise their trips to Banten Province's historical and cultural tourist destinations. As a result, information technology—mainly graph search algorithms—can be crucial for figuring out the best routes to take. To determine the shortest or most efficient path in a network of locations, algorithms like Depth-First Search (DFS) and Breadth-First Search (BFS) are frequently employed. There are numerous important advantages to figuring out the best path to Banten Province's archaeological tourism destinations. [2]Initially, travellers may navigate several sites more effectively, minimising confusion arising from the numerous attractions that need to be visited throughout various districts or subdistricts. Secondly, this algorithm can optimise travel routes, enabling travellers to ascertain the number of historical and cultural attractions they will visit. This is crucial for tourists with constrained visiting time who seek to optimise their experience. Moreover, graph search techniques can enhance the historical knowledge of travellers. Structured and optimised routes enable tourists to concentrate on acquiring and valuing the historical knowledge provided at each site. This technology enables archaeological site managers to offer more comprehensive and engaging travel guides using technology-based programs that deliver supplementary historical material, interactive maps, and optimal route recommendations. Consequently, the use of information technology through a graph search algorithm facilitates access and navigation in identifying the locations of historical-cultural tourist attractions in Banten Province, so enhancing the educational experience for visitors. This demonstrates that technology may significantly contribute to the preservation and comprehension of cultural material, rendering history more accessible and pleasant for present and future generations. This implementation enhances visit efficiency and guarantees that each visit is a thorough and significant learning experience. Numerous academics have utilised the BFS algorithm for identifying optimal paths, particularly in the tourist sector, as exemplified by Yosdarso Afero's 2022 study titled "BFS Algorithm Determines the Shortest Path in the Tourist City of Bukittinggi." This study demonstrated that the BFS algorithm can efficiently ascertain the shortest and most optimal route for tourists exploring various sites in Bukittinggi City. The results indicate that the algorithm can enhance travel time and distance, hence offering a better organised and efficient vacation experience.

[3]. In 2020, Julian Sahertian et al. conducted research applying the DFS algorithm in the tourism business. This implementation is executed on an Android-based tourism scheduling system in Trenggalek Regency to facilitate smart city initiatives. The application system was effectively installed according to the black box functional test outcomes, facilitating tour scheduling for tourists, particularly in Trenggalek Regency.[4] In addition to determining optimal travel routes, the BFS and DFS algorithms can be applied in various scenarios within information systems. For instance, research by Yuliana et al. in 2024, titled "Implementation of the DFS-BFS Algorithm in Accreditation Documents," demonstrates that the findings can enhance the speed and efficiency of searching for accreditation documents. Moreover, by employing a hybrid approach of the two algorithms, the system may swiftly and efficiently identify accreditation document storage paths in accordance with the established objectives.[5] The research concentrated on cooperative tree exploration, wherein robots navigate all edges of an unfamiliar tree and return to the root with efficiency. The review encompassed previous algorithms, notably the competitive method by Fraigniaud et al. and the performance metric by Brass et al., which ensured exploration durations in relation to offline expenses. The authors presented the Breadth-First Depth-Next (BFDN) algorithm, which demonstrates enhanced runtimes, surpassing earlier techniques and proving optimal for shallow trees. BFDN was enhanced to accommodate situations with restricted memory, hostile disruptions, and non-tree structures. A recursive variant of BFDN was introduced to enhance performance for deep trees, underscoring its versatility and efficacy in several contexts.

.[6] Two prominent algorithms for traversing trees and graphs are Breadth-First Search (BFS) and Depth-First Search (DFS). In the Neo4j implementation of the Cypher graph query language, Depth-First Search (DFS) was primarily utilised because of its more straightforward memory management, which could be efficiently handled by the Java Virtual Machine (JVM) when executed recursively. In contrast, BFS necessitated explicit data structures and utilised greater memory. Nevertheless, DFS was not consistently the superior option, as some real-world queries may have been enhanced by BFS. Permitting the choice between BFS or DFS for various segments of a query might have markedly enhanced the efficiency of Neo4j implementations. This thesis examined the efficacy of BFS and DFS in relation to different query forms in Cypher. It encompassed an empirical assessment of real enquiries and the development of a theoretical model to assist query planners in determining the most effective search method for particular query patterns. The research sought to address critical enquiries: whether enquiries exhibited superior performance with BFS compared to DFS in Cypher, under what circumstances BFS proved more cost-effective than DFS, and whether these findings could aid in the formulation of a theoretical model for query planning. [7] The efficiency and effectiveness of web page indexing in web crawler development are significantly influenced by the selection of the search algorithm. Two prevalent search algorithms in this context are breadth-first search (BFS) and depth-first search (DFS). The Breadth First Search algorithm is an expansive search technique that explores node n prior to advancing to node n+1.[8] This approach extracts nodes from the front of the queue subsequent to enqueuing the terminal nodes. The search will conclude if the node is the sought-after result. The search will encompass any nodes next to the node if it does not yield the desired result. The search is deemed complete when all nodes have successfully finished the verification stage and the queue is devoid of elements. The search resumes from the initial node in the queue. [9] Previous research has investigated the use of BFS and DFS in web crawler development. Some studies use BFS for even exploration at the same level, while DFS tends to unfold vertically to a certain depth before returning[10] This paper seeks to deliver an in-depth examination of graph traversal techniques, concentrating on Depth-First Search (DFS) and Breadth-First Search (BFS) as fundamental algorithms in network research. Both methods are essential in addressing significant issues in multiple fields, such as pathfinding, connection analysis, and data structure investigation. This study analyses the theoretical foundations, practical applications, and performance metrics of these algorithms in complex networks, providing insights into their operation under diverse scenarios. A comparative study of current studies will elucidate their respective advantages, limits, and prospective applications, providing a foundation for informed decision-making in the selection of traversal algorithms for network-related issues.

Background on graph traversal strategies.

The background on graph traversal strategies from the document discusses the fundamental approaches to navigating graph structures, particularly focusing on Breadth-First Search (BFS) and Depth-First Search (DFS). Here's a concise summary:

1. Graph Databases and Traversal:

   Graph databases organise data as nodes and edges, facilitating rapid exploration of relationships (e.g., identifying "friend-of-a-friend" connections). BFS and DFS serve as traversal methodologies, with BFS examining all adjacent nodes sequentially by level, whereas DFS delves extensively into pathways prior to backtracking.

2. Breadth-First Search (BFS):

   BFS examines all adjacent nodes of a vertex prior to progressing to the subsequent level.
   It is ideally suited for shortest-path computations and situations requiring extensive exploration.
   Variants such as Top-Down BFS and Bidirectional BFS improve efficiency by optimising the search direction or dividing workloads.

3. Depth-First Search (DFS):

   Depth-First Search (DFS) investigates a path to its maximum depth prior to backtracking.
   It is memory-efficient owing to its recursive nature, although it may be less effective for identifying shortest pathways or conducting extensive explorations.

4. Comparison and Applications:

   The selection between BFS and DFS is contingent upon the application, including memory limitations, data type, and particular query specifications.
   BFS necessitates explicit memory structures, rendering it more memory-intensive than DFS.[7]

- **Importance of DFS and BFS in computer science and network analysis.**

  Depth-First Search (DFS) and Breadth-First Search (BFS) are essential algorithms in computer science, especially in sophisticated algorithm design and network analysis. Recent studies and applications since 2018 have underscored their significance.

1. Advanced Algorithm Design

- **Space-Efficient Implementations**: Recent works have concentrated on enhancing the spatial complexity of Depth-First Search (DFS) and Breadth-First Search (BFS). Banerjee et al. (2016) devised linear-time methods for BFS and DFS that function under constrained space conditions, hence improving their utility in memory-restricted settings

- **In-Place Graph Algorithms**: Chakraborty et al. (2017) Established frameworks for the creation of in-place graph algorithms, facilitating efficient depth-first search (DFS) and breadth-first search (BFS) traversals with little additional space requirements. This development is vital for extensive graph processing when memory optimisation is needed.

2. Network Analysis

- **Temporal Graph Neural Networks**: The incorporation of DFS into temporal graph neural networks has been investigated to identify dynamic patterns in temporal graphs. Singer et al. (2022) introduced tBDFS, an innovative temporal GNN architecture that utilises DFS for effective information aggregation in temporal routes, exhibiting enhanced performance in link prediction test.

- **Graph Compression Techniques**: Efficient graph traversal techniques, such as BFS, are fundamental to graph compression algorithms, which are crucial for the analysis of large-scale networks. Besta and Hoefler (2019) conducted an extensive survey on lossless graph compression methods, emphasising the significance of BFS in attaining space-efficient graph representations.

3. Parallel Computing

- **Parallel BFS Algorithms**: The advancement of parallel BFS algorithms has been crucial for managing large graphs in distributed systems. Recent improvements encompass optimisation algorithms for load balancing and data structures that improve the efficiency of parallel BFS implementations.

4. Artificial Intelligence

- **Heuristic Search Strategies**: Depth-First Search (DFS) and Breadth-First Search (BFS) underpin numerous heuristic search methodologies in artificial intelligence. Everitt and Hutter (2015) performed analytical investigations on the selection dilemma between BFS and DFS, offering insights into their performance across various search circumstances.

- **5. Educational Resources**

- **Algorithm Tutorials** Educational platforms have revised their content to incorporate the most recent uses and enhancements of DFS and BFS. GeeksforGeeks offers tutorials on the uses, benefits, and drawbacks of DFS and BFS, providing practical insights into their implementation.[11], [12], [13], [14]

- Commonly employed for pathfinding and route optimisation, Depth-First Search (DFS) and Breadth-First Search (BFS) are cornerstone algorithms in computer science and network research. When it comes to unweighted graphs, BFS is great at discovering the shortest pathways, but DFS is great at investigating every potential path and making sure coverage is maximised. They are useful in tourist route planning and cultural site management because they help with decision-making by balancing location coverage with distance efficiency. While DFS is more suited to thorough searches, BFS aims to minimise travel distance. Both techniques are applicable in both theoretical and practical settings; for example, they are used in information systems for document retrieval and network modelling.[15]
- Commonly employed in temporal data mining and frequent episode finding, Depth-First Search (DFS) and Breadth-First Search (BFS) are cornerstone algorithms in computer science and network analysis. While DFS is more typically employed for pattern-growth tasks and deeper data investigation, BFS is appropriate for large datasets because to its efficiency in candidate creation and frequency counts. Finding shortest pathways and analysing hierarchical structures are areas where BFS really shines, while exhaustive searches in complicated networks are where DFS really shines. Both algorithms are essential for improving data analysis in many contexts by spotting trends, cutting down on unnecessary information, and bolstering resource management.[16]

2. Fundamentals of Graph Traversal

Definition of graph traversal.

When exploring, analysing, or processing data, graph traversal is the process of methodically traversing all the nodes (or vertices) in a graph structure. To get from one node to another, traversal makes use of the graph's edges. Typical approaches to traversal:

1. **Depth-First Search (DFS):** Explores as far down a branch as possible before backtracking.
2. **Breadth-First Search (BFS):** Explores all neighbors of a node before moving to the next level.
   Applications such as route discovery, social network research, network analysis, and many more rely on traversal.[7]

Key Differences Between DFS and BFS

DFS and BFS have distinct approaches to traversal and serve different purposes in various applications. BFS systematically investigates nodes in a sequential manner using a queue, which makes it particularly effective for identifying the shortest paths and detecting connected components. DFS employs a stack or recursion to delve deeply into a problem before backtracking, frequently utilised in tasks such as cycle detection, topological sorting, and planarity testing. BFS builds a tree that prioritises the first elements encountered, whereas DFS develops a tree that emphasises the last elements encountered. While both algorithms typically enable linear-time tree recognition, specific tasks such as the "last-in" tree for BFS and the "first-in" tree for DFS are NP-complete. BFS is ideal for surface-level searches, whereas DFS excels in tasks requiring deeper exploration.[17]

The traversal strategies and use cases of DFS and BFS are fundamentally different. BFS systematically investigates nodes in layers through a queue, which makes it particularly effective for finding the shortest path and managing network routing. In contrast, DFS delves deeply into nodes using a stack or recursion, allowing for thorough exploration before backtracking, making it well-suited for tasks like cycle detection and topological sorting. BFS utilises a significant amount of memory but proves to be effective for unweighted graphs, whereas DFS is more memory-efficient and performs exceptionally well in tasks that require deep exploration. BFS prioritises breadth, while DFS concentrates on depth, rendering each method appropriate for various graph-related challenges.[18]

Applications of DFS and BFS in Complex Networks

DFS and BFS are extensively utilised in intricate networks, particularly in the realm of temporal data mining, as they assist in uncovering frequent patterns that involve concurrent events. BFS demonstrates notable effectiveness in the analysis of financial data, sensor networks, and transaction logs, owing to its capability to efficiently manage large datasets and concurrent events. This method is utilised in stock market analysis to detect simultaneous price movements and in sensor networks to identify spatio-temporal correlations. Furthermore, BFS is instrumental in transaction analysis as it identifies temporal patterns in customer behaviour. In contrast to DFS, BFS tends to demonstrate greater efficiency in situations characterised by simultaneous events, primarily due to its reduced memory usage and quicker processing capabilities.[16] DFS and BFS play a crucial role in the analysis of complex networks, especially within graph databases such as Neo4j, where they contribute significantly to query optimisation. BFS proves to be efficient for smaller graphs and shorter queries, whereas DFS excels in managing deeper searches with improved memory utilisation. BFS is commonly utilised for identifying the shortest paths in unweighted graphs and for investigating connections within social networks. Both algorithms hold significant importance in evaluating graph algorithms concerning their runtime and memory efficiency. Advanced implementations, such as Multi-Source BFS and Concurrent BFS on GPUs, significantly improve performance in extensive networks. In summary, BFS and DFS play a vital role in data analysis, query optimisation, and effective network processing.[19] DFS and BFS play a crucial role in intricate networks for pathfinding, route optimisation, and network analysis. BFS is highly effective at identifying the shortest path in unweighted graphs, whereas DFS proves to be advantageous for exploring deeper paths. These applications find extensive use in network routing, such as with the OSPF protocol, geographic mapping for GPS systems, search engine indexing, and peer-to-peer networks like BitTorrent. Furthermore, BFS and DFS serve as important educational resources for visualising algorithms, enhancing comprehension of graph traversal and shortest path methods. Their adaptability is essential for enhancing efficiency and addressing challenges in multiple fields.[20] DFS and BFS play a vital role in the analysis of complex networks, offering a range of applications. BFS proves to be efficient in detecting connected components, testing for bipartiteness, and computing shortest paths, which makes it appropriate for tasks related to network segmentation and routing. DFS proves to be instrumental in pinpointing strongly and biconnected components, facilitating topological sorting, and conducting planarity testing, which supports endeavours such as network reliability, task scheduling, and circuit design. Collectively, these algorithms are essential for navigating paths, exploring structures, and addressing graph-related challenges in various fields.[17] DFS and BFS play a crucial role in intricate networks, commonly applied in link prediction, social network analysis, and studies in the biomedical field. They play a crucial role in estimating the likelihood of connections between nodes, forecasting social interactions, and enabling the extraction of knowledge in the fields of bioinformatics and medical research. Furthermore, both algorithms facilitate the analysis of graph structures, improve machine learning models via feature extraction, and assist in modelling intricate systems such as labour markets and longitudinal datasets, showcasing their adaptability across multiple fields.[18]

Comparative Analysis of Depth-First Search (DFS) and Breadth-First Search (BFS) in Complex Network Applications

The table below provides a comparative analysis of Depth-First Search (DFS) and Breadth-First Search (BFS) algorithms across various applications in complex networks. The two essential strategies for graph traversal have found extensive application across multiple fields, such as pathfinding, network flow optimisation, collaborative exploration, and decision-making processes. The table meticulously outlines essential elements of each algorithm's execution and effectiveness by summarising the datasets utilised, metrics assessed, network types examined, and findings documented in earlier research. This table provides a detailed comparison of DFS and BFS, emphasising their unique strengths, limitations, and the ways in which they intersect or enhance each other. The findings presented enhance comprehension of the performance of these algorithms across various network configurations and problem limitations, acting as a crucial guide for choosing suitable traversal methods in intricate network evaluations.

Table 1 : A detailed comparison of DFS and BFS, insights from different literature

Article ID	Algorithm Analyzed	Depth-First Search (DFS)	Breadth-First Search (BFS)	Dataset Used	Metrics Evaluated	Network Type	DFS Performance	BFS Performance	Observations/Insights
[21]	Max-Flow on Extended Networks	Optimised to identify a novel augmenting path in residual networks for maximum flow.	Utilised a conventional method to identify augmenting pathways and maximum flow.	Residual network, Extended network model	Max-flow value, Algorithm complexity	Extended Network (weighted, directed)	O(2)	E	+ 2
[22]	Breadth-First Depth-Next (BFDN)	Utilised for solitary robotic exploration and optimum tree traversal.	Fundamental aspect of the BFDN algorithm for collaborative tree exploration	Unknown trees with N nodes, D depth	Exploration time (rounds), computational efficiency	Trees, collaborative exploration	Not explicitly utilised in this algorithm	O(2n/k+D2log(k)) for collaborative exploratio	BFDN enhances the speed and scalability of tree exploration by adapting BFS and DFS methods for use with many robots. Takes into account limitations such as memory space and hostile environments
[23]	Tree-Based Tracking for Cartesian Coordinates	Not applied in this study.	Iteratively investigates each node until a solution is discovered.	Cartesian field coordinates	Efficiency in processing, memory utilisation, and optimal paths	Cartesian grid traversal	Not evaluated.	By investigating every conceivable answer, BFS ensures the shortest path. High memory utilisation from storing all the nodes is one of the drawbacks.	When it comes to grid-based problems, BFS is reliable for discovering optimal paths, but when it comes to deeper or broader levels, it requires a lot of memory and time.
[24]	Route selection in MANET	Improves performance, optimises packet delivery, and decreases	Cuts down on power usage and wait times.	MANET with varying nodes (10-100 nodes)	Throughput, energy usage, packet loss, and packet delivery ratio (PDR)	Dynamic wireless topology	The throughput was 70.67 Kbps, and the PDR was 86.38%. Comparing to BFS, there is a decrease in packet loss of 16.82%.	- Better delay (113,228 ms avg) and energy efficiency (0.46 J avg). \n- Slightly lower PDR (79.86%) and throughput (68.14 Kbps).	In situations where precision and dependability are paramount, DFS is the way to go, whereas BFS is the way to go for low-delay, energy-efficient routing.
[25]	Retrosynthesis Planning	Our main goal is to identify the most probable pathways for synthesis using prioritised reactions.	When trying to cover a lot of ground in a chemical search, scalability is usually an issue.	USPTO Reaction Dataset	Efficacy in route planning, success rate, and route quality (length, cost).	AND-OR Tree for retrosynthesis	Works well for synthesis problems with large costs; optimises particular processes better.	Easily scalable, but not as precise when assessing AND-OR search tree substructures.	Retro* (a hybrid, A*-like neural-guided BFS and DFS approach) outperforms both traditional BFS and DFS in retrosynthesis planning.

[26]	Branch-and-Bound Algorithms	Used for delving extensively into one branch before retracing one's steps.	Initially investigates all nodes on the same level of the tree.	Various NP-hard problem	Data storage, processing speed, and investigational breadth	Optimization problems with tree structures	Efficient for bounded-depth problems, uses little memory.	Verifies that the best solution will be found even without pruning	Even while DFS uses little memory, it may not find the best paths if it explores less-than-ideal ones first. Although it uses a lot of memory, BFS methodically discovers the best solution in search areas that have no bounds.
[27]	RAW-GNN	Discovers heterophily data by delving into more complex graph structures.	Identifies homophily by zeroing in on neighbouring nodes	Homophily and heterophily databases (such as Citeseer, PubMed, and Cora)	Efficiency during runtime, accuracy of node categorisation, and feature aggregation	Heterophily and homophily graphs	Shows improved accuracy and resilience when run on datasets that are dominated by heterophily.	Provides somewhat improved classification performance; appropriate for homophily-dominant datasets.	The state-of-the-art results on heterophily and homophily graphs are produced by combining BFS and DFS using random walk techniques, which allows for the handling of different graph architectures.
[28]	ATM Search in Padang Sidempuan	Continues to search along a single branch until it finds a way to the destination.	This study does not analyse it, but it compares it to the Greedy algorithm.	Weighted graph representing ATM locations	Time to execution (ms), algorithm complexity, and distance to ATM	Weighted directed graph	Average running time: 239.97 ms; average distance: 3033.56 meters	Not relevant; On average, the greedy algorithm covered 2035.26 meters in 274.85 milliseconds.	DFS exhibits greater time efficiency, yet it is less optimal in terms of distance when compared to the Greedy algorithm for locating the nearest ATM. The greedy algorithm results in shorter paths; however, it incurs higher computational costs.
[29]	Disease Diagnosis in Areca Plants	Investigates routes to all symptoms for disease diagnosis using a depth-first approach.	Conducts a systematic search, progressing through each level to ensure comprehensive consideration of all symptoms.	16 diseases and 33 symptoms of Areca plants	Diagnostic accuracy, computation time, completeness	Decision tree graph	Appropriate for an in-depth analysis of complex symptom hierarchies.	Facilitates comprehensive examination of symptoms at every level, minimising the likelihood of oversight.	BFS is more effective for methodically verifying all symptoms. DFS is more expedient but may not evaluate other options as thoroughly.
[20]	Pathfinding Visualizer	Investigates a single trajectory thoroughly before retracing steps to examine alternative avenues.	Investigates all adjacent nodes at a certain level prior to delving deeper into the network.	Grid-based graphs for visual aid	Execution duration, path efficiency, comprehensibility	Static and grid-based graphs	It took 12.65 seconds to run and gives full traversal for deeper paths.	It took 6.46 seconds to run and found the shortest route quickly.	In this case, BFS is better at finding the shortest routes. DFS is slower, but it can help you find complicated or deep routes. Both of them are shown graphically well.
[30]	BFS and DFS in Graph Traversal	It uses less memory than BFS and follows one path until it	Ensures full graph exploration and shortest path discovery by	Simulated grid maps of varying sizes (50×50 to 200×200)	Number of nodes visited, runtime efficiency, path length	Static weighted graph	Optimal for investigating unrelated parts; slower in big graphs owing to	Efficient in guaranteeing shortest pathways, but keeping all border nodes	In well-structured environments, BFS finds the shortest pathways, while in less-structured graphs, DFS finds the most efficient use of space.

		hits a dead end, then it turns around.	exploring all neighbours level by level.				comprehensive search of deep paths.	demands more memory.	
[16]	Frequent Episode Mining	Due to the large memory requirement of maintaining occurrence timestamps, it is inefficient for simultaneous events.	Unique candidates and frequency counting make it efficient for serial episodes with numerous occurrences at once.	Artificial datasets featuring a range of noise intensities	Runtime, memory usage, frequency accuracy	Temporal graphs/sequences	Efficient in small datasets but has trouble handling big ones.	Compared to DFS methods, it achieves faster runtime and improved scalability for datasets with significant noise.	When it comes to performance and scalability, BFS-FA shines over DFS, particularly on noisy datasets. While BFS-FA offers superior performance, DFS-HUE is only able to deliver average results.
[31]	Maze-solving algorithms	Explores one road to the deepest level before backing up to consider others.	Examines all nodes level-by-level to analyse all pathways	25x25 maze grids with random obstacles	Path length, runtime	Grid-based mazes	Shorter execution time than BFS, longer path.	It takes longer than DFS but finds shorter pathways.	DFS is fast but may miss the shortest path. BFS finds and guarantees the shortest path in large mazes but takes longer.
[32]	Graph Traversal Algorithms	Takes one path seriously before turning around.	By-level exploration of all nodes.	Possible unweighted graphs	Process duration, memory use, path length	Static graph structures	Apt for issues calling for extensive investigation; nonetheless, it may fail to reveal the shortest routes.	Ensures the quickest path in graphs without weights; big graphs cause memory usage to spike.	The best method for shortest path identification is BFS, while DFS is preferable for space-constrained situations or deep graph exploration with multiple branches.
[33]	Graph-based Path Planning	Fully explores one path before returning.	Explores same-level nodes before	Obstacle-simulated graphs	Path length, time complexity, heuristic cost	Directed, weighted graphs	Good at finding deeper, particular paths.	Guarantees shortest path but uses more RAM.	DFS is efficient for specialised deep searches, while BFS finds the shortest path in unweighted graphs but takes longer in bigger graphs.
[34]	Network Sampling Strategies	Risks local biases by deeply exploring a single path under API limits.	Level-by-level exploration of all nodes ensures network sampling.	Simulated synthetic graphs and APIs	API cost efficiency, sampling precision, representativeness	Undirected and directed networks	Works well in deeper exploration scenarios but is expensive under API limits.	Performs reliably but uses more memory with huge graphs.	BFS covers larger graphs and is more representative, while DFS is memory-efficient but may miss essential network features given API limits.
[15]	Route Optimization for Tourism	Backtracking after thorough exploration optimises visits to more places.	Optimises distance by exploring neighbouring areas first.	Tourist attractions in Banten Province graph	Travel distance, tourist sites visited	Weighted graph of locations	Maximises tourist destinations but increases journey time.	Guarantees shorter travel routes but may visit fewer places.	DFS improves tourist experiences by increasing site visitation. BFS is preferable for travellers that value shorter, faster travel.

[35]	Average Sensitivity in Grids	Not directly examined, yet recognised for extensive exploration prior to retracing steps.	Examines all nodes systematically, layer by layer; stability in response to disturbances is assessed.	Grids (e.g., 5×5, m×n lattices	Mean sensitivity, resilience, stability amidst variations	Grid graphs	Not detailed for sensitivity measures	Sensitivity demonstrated to be O(1) in some configurations.	The average sensitivity of BFS on grid graphs is considerably lower (O(1)) than that in arbitrary graphs (Θ).
[36]	Swarm Intelligence Search	Delves into a singular route thoroughly prior to retracing steps; ineffective in unfamiliar landscapes.	Examines all nodes systematically by level; ensures the shortest path, although is computationally intensive.	Stochastically produced graphs and authentic landscapes	Path length, cost (time and agents), scalability	Randomised and physical topographical graphs	Exhibits proficiency in deterministic environments but encounters difficulties in unfamiliar contexts and demonstrates limited scalability.	Ensures optimal pathways but results in elevated expenses and agent utilisation in expansive terrains.	BFS is optimal for compact or organised terrains where precision is essential. DFS encounters difficulties in unfamiliar situations because of local traps, while it remains computationally efficient. Swarm methodologies enhance both aspects.
[37]	Shortest Path in Neutrosophic Environment	Not assessed; BFS is the principal emphasis.	Augmented to accommodate neutrosophic numbers denoting ambiguous edge weights.	Neutrosophic graph representations	Path length, computational efficiency, algorithmic stability	Weighted, connected graphs	Not analyzed in this study.	Attains consistent outcomes with 95%-100% precision in determining the shortest pathways despite ambiguity.	BFS demonstrates significant efficacy in uncertain contexts utilising neutrosophic parameters; nonetheless, its computing efficiency is contingent upon the size and complexity of the graph.
[38]	Iterative Depth-First Search (IDFS)	Conducts an in-depth exploration within a constrained framework, effectively managing non-deterministic states in FOND planning.	Principles of BFS are implicitly contrasted, although they are not the main emphasis of this work.	Comparison of the IPC-FOND and NEW-FOND databases	Time spent planning, scope, number of solution revisions, and policy size	Fully Observable Non-Deterministic (FOND) graphs	Obtains strong cyclic policies efficiently and solves 422 tasks with resilient performance under limitations.	Indirect comparison reveals that newer FOND benchmarks are less efficient for BFS-based methods like FONDSAT.	When compared to planners that are influenced by BFS, such as PRP and FONDSAT, IDFS offers better handling of depth-bounded cyclic policies and competitive planning time and coverage.
[39]	Breadth-First Depth-Next (BFDN)	Replicates DFS by having robots explore subtrees and eventually find their way back to the root.	Makes use of BFS to distribute robot exploration evenly by sending them to previously unexplored edges.	Virtual trees with different number of nodes (n) and depth (D)	Phases of exploration, scalability with k agents	Advanced graph models and trees	Parts similar to DFS guarantee coverage of all subtrees but may be less efficient when it comes to retracing.	The efficient assignment of paths in sparse tree areas reduces duplication in systems with several agents.	Integrates BFS and DFS ideas to ensure efficient collaborative exploration with runtime and scalability guarantees for both tree depth and breadth.

[40]	Graph Sampling for Classification	Embedding with a combination of DFS and BFS techniques (local/global view balancing) is achieved using node2vec.	Makes use of DeepWalk, a random walk algorithm that prioritises wide-area discovery of local connections.	Synthetic (BA, WS) and real-world datasets	Precision in classification, capacity for feature extraction	Directed and weighted graphs	Decently finds depth-centric characteristics in networks that don't require scaling.	Does a great job of identifying small-world network clustering and community traits	Both BFS and DFS function well together in convolutional neural networks to achieve high classification accuracy, and they balance exploration for robust network representation.
[41]	Graph Traversal Algorithms	Investigates one avenue thoroughly before reversing course to investigate other possibilities .	Investigates each node on the surface before delving further.	Network datasets based on theory and practice	Coverage of paths, processing time, and memory consumption	Directed and undirected graphs	Apt for acyclic graph cycle detection and topological sorting	In unweighted graphs, it guarantees full exploration and finds the shortest path.	In contrast to BFS's robustness in guaranteeing thorough graph exploration and shortest paths, DFS is space-efficient and excels at problem-specific tasks.

9. Conclusion

This paper offers a thorough comparative examination of Depth-First Search (DFS) and Breadth-First Search (BFS) algorithms, highlighting their theoretical foundations, performance metrics, and applications in various complex network contexts. The investigation demonstrated that BFS typically surpasses DFS in tasks necessitating shortest path identification and thorough coverage, rendering it especially appropriate for unweighted graphs and network configurations where extensive exploration is critical. In contrast, DFS exhibited benefits in situations necessitating extensive investigation with minimal memory usage, notably beneficial for decision trees, hierarchical datasets, and issues with constrained depth.

The comparative analysis of current studies underscored the significance of choosing traversal algorithms according to the dataset type, network architecture, and performance criteria. BFS is more efficient in grid-based and structured graphs, whereas DFS excels in scenarios where memory efficiency is paramount. Moreover, hybrid methodologies such as Breadth-First Depth-Next (BFDN) and Pruned-BFS have demonstrated the capacity to integrate the advantages of both methods, hence improving performance and scalability in certain applications such as collaborative exploration and network flow optimisation.

This paper emphasises that the selection between DFS and BFS should be determined by the particular requirements of the problem domain, taking into account issues such as scalability, optimality, and resource limitations. Future study may investigate additional enhancements of hybrid methodologies, broadening their applicability to dynamic and extensive complex networks while persistently assessing their efficacy in relation to emerging graph traversal issues.

10. References

[1] D. Negri Wijaya, I. Lutfi, R. R. Hudiyanto, D. Y. Wahyudi, and F. Ariska, “Daya Negri Wijaya, Daya Tarik Wisata sejarah budaya di Malang… Daya tarik wisata sejarah budaya di Malang Raya.”

[2] W. L. Putri and N. Jarti, “Algoritma General and Test Menggunakan Metode Depth First Search Dalam Penentuan Jalur Rute Terpendek,” *Brahmana: Jurnal Penerapan Kecerdasan Buatan*, vol. 4, no. 2, pp. 154–163, 2023.

[3] Y. Afero *et al.*, “ALGORITMA BEST FIRST SEARCH MENENTUKAN LINTASAN JALUR TERPENDEK PADA KOTA WISATA BUKITTINGGI,” *JOISIE Journal Of Information System And Informatics Engineering*, vol. 5, no. Desember, pp. 138–145, 2021.

[4] J. Sahertian, M. A. D. Widyadara, and F. R. Agista, “IMPLEMENTASI SISTEM PENJADWALAN WISATA DI KABUPATEN TRENGGALEK BERBASIS ANDROID UNTUK MENUNJANG SMART CITY,” *Joutica*, vol. 5, no. 1, p. 326, Mar. 2020, doi: 10.30736/jti.v5i1.336.

[5] M. Qulub and I. Shanti Bhuana, “IMPLEMENTASI ALGORITMA DEPTH-FIRST SEARCH DAN BREADTH-FIRST SEARCH PADA DOKUMEN AKREDITASI,” 2024. [Online]. Available: http://jurnal.goretanpena.com/index.php/JSSR

[6] R. Cosson, L. Massoulié, and L. Viennot, “Breadth-First Depth-Next: Optimal Collaborative Exploration of Trees with Low Diameter,” Jan. 2023, [Online]. Available: http://arxiv.org/abs/2301.13307

[7] A. Olsson and T. Magnusson, “Implementing and Evaluating a Breadth-First Search in Cypher”.

[8] A. Muhardono, “Penerapan Algoritma Breadth First Search dan Depth First Search pada Game Angka,” *Jurnal Minfo Polgan*, vol. 12, no. 1, pp. 171–182, Mar. 2023, doi: 10.33395/jmp.v12i1.12340.

[9] A. Mustaqim, D. B. Dinova, M. S. Fadhilah, R. Seivany, B. Prasetiyo, and M. A. Muslim, “Optimizing the Implementation of the BFS and DFS algorithms using the web crawler method on the kumparan site,” *Journal of Soft Computing Exploration*, vol. 5, no. 2, pp. 200–206, Jul. 2024, doi: 10.52465/joscex.v5i2.309.

[10] M. Parmar and H. J. Kaur, “Comparative analysis of secured hash algorithms for blockchain technology and internet of things,” *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 3, 2021.

[11] S. Chakraborty, A. Mukherjee, V. Raman, and S. R. Satti, “Frameworks for Designing In-place Graph Algorithms,” Nov. 2017, [Online]. Available: http://arxiv.org/abs/1711.09859

[12] U. Singer, H. Roitman, I. Guy, and K. Radinsky, “tBDFS: Temporal Graph Neural Network Leveraging DFS,” Jun. 2022, [Online]. Available: http://arxiv.org/abs/2206.05692

[13] T. Everitt and M. Hutter, “A topological approach to meta-heuristics: analytical results on the BFS vs. DFS algorithm selection problem,” *arXiv preprint arXiv:1509.02709*, 2015.

[14] N. Banerjee, S. Chakraborty, V. Raman, and S. R. Satti, “Improved Space efficient linear time algorithms for BFS, DFS and applications,” Jun. 2016, [Online]. Available: http://arxiv.org/abs/1606.04718

[15] Mochammad Darip, Sigit Auliana, A. K. Anam, Parimin, and Anugerah Agung, “Comparison of BFS and DFS Algorithm for Routes to Historical-Cultural Tourism Locations in Banten Province,” *Journal of Advances in Information and Industrial Technology*, vol. 6, no. 2, pp. 113–122, Oct. 2024, doi: 10.52435/jaiit.v6i2.560.

[16] S. B. Gandreti, A. Ibrahim, and P. S. Sastry, “Breadth-First Search Approach for Mining Serial Episodes with Simultaneous Events,” in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Jan. 2024, pp. 36–44. doi: 10.1145/3632410.3632445.

[17] R. Scheffler, “On the recognition of search trees generated by BFS and DFS,” *Theor Comput Sci*, vol. 936, pp. 116–128, Nov. 2022, doi: 10.1016/j.tcs.2022.09.018.

[18] J. Dörpinghaus, T. Hübenthal, and D. Stepanov, “A novel DFS/BFS approach towards link prediction,” Sep. 2024, [Online]. Available: http://arxiv.org/abs/2409.11687

[19] A. Olsson and T. Magnusson, “Implementing and Evaluating a Breadth-First Search in Cypher”.

[20] H. Pandey, A. Kumar, and S. Verma, “Pathfinding Visualizer Using Multiple Graph Algorithms,” *International Journal of Research and Analytical Reviews*, 2023, [Online]. Available: www.ijrar.org

[21] T. Ngoc Viet, H. LE Minh, T. M. Kim Van, T. Hung Anh, N. Tuyen Linh, and H. Daxue Xuebao, “IMPROVE THE POWER OF FORD FULKERSON ALGORITHM AND DEPTH FIRST SEARCH,” *Journal of Hunan University Natural Sciences*, doi: 10.17605/OSF.IO/XWVN3.

[22] R. Cosson, L. Massoulié, and L. Viennot, “Breadth-First Depth-Next: Optimal Collaborative Exploration of Trees with Low Diameter,” Jan. 2023, [Online]. Available: http://arxiv.org/abs/2301.13307

[23] J. Sihotang, “Analysis Of Shortest Path Determination By Utilizing Breadth First Search Algorithm.” [Online]. Available: http://ejournal.seaninstitute.or.id/index.php/InfoSains

[24] Alamsyah, A. Amir, M. Subito, R. Fauzi, and Amirullah, “Performance analysis of breadth-first search and depth-first search on MANET for health monitoring system,” in *IOP Conference Series: Earth and Environmental Science*, Institute of Physics, 2022. doi: 10.1088/1755-1315/1075/1/012011.

[25] B. Chen, C. Li, H. Dai, and L. Song, “Retro*: Learning Retrosynthetic Planning with Neural Guided A* Search,” 2020. [Online]. Available: https://github.com/binghong-ml/

[26] D. Morrison, S. H. Jacobson, E. Sewell, D. R. Morrison, J. J. Sauppe, and E. C. Sewell, “Branch-and-Bound Algorithms: Recent Advances in Searching, Branching, and Pruning.” [Online]. Available: https://www.researchgate.net/publication/376889863

[27] D. Jin *et al.*, “RAW-GNN: RAndom Walk Aggregation based Graph Neural Network,” Jun. 2022, [Online]. Available: http://arxiv.org/abs/2206.13953

[28] D. Rachmawati, S. Efendi, and A. S. Situmorang, "COMPARATIVE ANALYSIS OF DEPTH-FIRST SEARCH ALGORITHM AND GREEDY ALGORITHM AT NEAREST ATM SEARCH IN PADANG SIDEMPUAN CITY," *J Theor Appl Inf Technol*, vol. 15, p. 17, 2020, [Online]. Available: www.jatit.org

[29] F. J. Pane, E. Rianti, and H. Marfalino, "Application of an Expert System with the Breadth First Search (BFS) Method in Diagnosing Areca Plant Diseases," *Journal of Computer Scine and Information Technology*, pp. 55–59, Apr. 2024, doi: 10.35134/jcsitech.v10i2.102.

[30] H. Wang, S. Lou, J. Jing, Y. Wang, W. Liu, and T. Liu, "The EBS-A* algorithm: An improved A* algorithm for path planning," *PLoS One*, vol. 17, no. 2 February, Feb. 2022, doi: 10.1371/journal.pone.0263841.

[31] I. P. Chinemerem, "A COMPREHENSIVE AND COMPARATIVE STUDY OF DFS, BFS, AND A* SEARCH ALGORITHMS IN A SOLVING THE MAZE TRANSVERSAL PROBLEM ☆," 2022.

[32] J. Iyanda, "Title : A Comparative Analysis of Breadth First Search (BFS) and Depth First Search (DFS) Algorithms." [Online]. Available: https://www.researchgate.net/publication/370751322

[33] L. Shi, "Research on Path Planning Method based on Graph Search Algorithm," 2024.

[34] Naoki. Abe, *2018 IEEE International Conference on Big Data : proceedings : Dec 10 - Dec 13, 2018, Seattle, WA, USA*. IEEE, 2018.

[35] M. Assari, "Average Sensitivity of Breadth-First Search Algorithm on Grids," 2019.

[36] *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2019.

[37] P. K. Raut, S. P. Behera, S. Broumi, and A. Baral, "Evaluation of Shortest Path by using Breadth-First Algorithm under Neutrosophic Environment," *HyperSoft Set Methods in Engineering*, vol. 1, pp. 34–45, Jan. 2024, doi: 10.61356/j.hsse.2024.18350.

[38] R. F. Pereira, A. G. Pereira, F. Messa, and G. De Giacomo, "Iterative Depth-First Search for Fully Observable Non-Deterministic Planning," Apr. 2022, [Online]. Available: http://arxiv.org/abs/2204.04322

[39] R. Cosson, L. Massoulié, and L. Viennot, "Efficient Collaborative Tree Exploration with Breadth-First Depth-Next," 2023, doi: 10.4230/LIPIcs.DISC.2023.14ï.

[40] S. M. Arul, G. Senthil, S. Jayasudha, A. Alkhayyat, K. Azam, and R. Elangovan, "Graph Theory and Algorithms for Network Analysis," in *E3S Web of Conferences*, EDP Sciences, Jul. 2023. doi: 10.1051/e3sconf/202339908002.

[41] R. Xin, J. Zhang, and Y. Shao, "Complex Network Classification with Convolutional Neural Network," 2020. [Online]. Available: https://unstats.un.org/unsd/trade/sitcrev4.htm