Extending Degree Based Search Algorithm for All Directed Disconnected Graphs

Shyma Puthiyaveetil and Sanil Shanker Kuniyapoil

ABSTRACT

Graph traversal is a fundamental process in numerous applications, including citation analysis, influence maximization, and recommendation systems. Standard graph traversal methods—Breadth-First Search (BFS) and Depth-First Search (DFS)—exhibit performance limitations when applied to large, unconnected, or directed graphs, particularly when the objective is to identify influential nodes efficiently. In this research, we present a novel graph traversal strategy termed Degree-Based Search (DBS), which prioritizes nodes in descending order of degree values to enable early discovery of high-impact nodes. We analyze the computational time complexity of the proposed method as O((|V | + |E|) log |V |), where |V | and |E| denote the number of vertices and edges in the graph, respectively. Our empirical evaluation on the Cora citation dataset shows that DBS-1.00 outperformed BFS and DFS by identifying the top 10 most cited nodes with only 45.6% graph traversal and locating the top 20 nodes at an average position of 13.3%, compared to 64.3% and over 23% for BFS and DFS, respectively. Performance improvement is especially significant in applications such as biological networks and social media graphs, where high-degree nodes are key hubs of information.

Keywords: Degree-based search algorithm, Directed disconnected graphs, Node degree prioritization, Influential node discovery, Citation network traversal

Department of Information Technology Kannur University, Kerala, India ROR

Correspondence to:
Shyma Puthiyaveetil,
shymapv@kannuruniv.ac.in

Additional material is published online only. To view please visit the journal online.

Cite this as: Shyma PV and Shanker SKP. Extending Degree Based Search Algorithm for All Directed Disconnected Graphs. Premier Journal of Science 2025;15:100203

DOI: https://doi.org/10.70389/PJS.100203

Peer Review

Received: 12 August 2025

Last revised: 24 August 2025

Accepted: 17 December 2025

Version accepted: 1

Published: 16 January 2026

Ethical approval: N/a

Consent: N/a

Funding: N/a

Conflicts of interest: N/a

Author contribution:
Shyma Puthiyaveetil and Sanil Shanker Kuniyapoil – Conceptualization, Writing – original draft, review and editing

Guarantor: Shyma Puthiyaveetil

Introduction

Understanding the robustness and structural behaviour of complex systems, particularly in situations involving social, biological, and communication networks, depends critically on the analysis of connectivity in directed graphs. The necessity for effective traversal strategies catered to disconnected and direction-sensitive structures is still poorly understood, despite the fact that conventional connectivity metrics like edge connectivity, vertex connectivity, and strongly connected components have been thoroughly investigated. Degree-based search strategies, which prioritize nodes according to their degree properties to improve performance in both connected and sparse environments, have recently become a popular paradigm for graph traversal. However, current degree-based algorithms are limited in their applicability to real-world networks, which frequently display fragmentation and asymmetry in reachability, because they primarily assume connectedness and are ill-suited to handle general directed disconnected graphs. An extended degree-based search algorithm created especially for all classes of directed disconnected graphs is presented in this paper to fill this gap. By analysing the effects of various edge combinations each carrying varying degrees of truth, uncertainty, and falsity on the network's overall structure, edge connectivity in neutrosophic graphs is ascertained. The method finds the fewest and most significant edge sets that jeopardize connectivity by methodically eliminating subsets of edges and utilizing graph traversal techniques such as DFS or BFS to check for disconnection. This approach takes into consideration edges' structural function as well as the inherent ambiguity of their existence.[1,2] The extended algorithm serves as a unified traversal strategy capable of handling strong, unilateral, and weak connectivity in directed disconnected graphs. It uses degree-based prioritization to dynamically process high- and low-degree nodes for improved reachability and incorporates Tarjan's depth-first search to accurately detect strongly connected components.[3]

Related Work

Academic research has noted a considerable level of interest in the use of node degrees as a core requirement for graph traversal. High-degree nodes are the central hubs that significantly determine connectivity and information flow in different networks. The paper, "Degree-aware Hybrid Graph Traversal on FPGA-HMC Platform," introduces an innovative graph traversal method that balances top-down and bottom-up approaches to traverse more efficiently.[4] The algorithm first applies a top-down approach, traversing from a source vertex and visiting its neighbours. This is efficient for sparse frontiers. As graph traversal progresses, the algorithm switches to a bottom-up approach when the frontier becomes huge and the top-down approach gets slow. This switching between these two approaches is controlled by two thresholds, α and β, by frontier size and number of edges, to allow the algorithm to adjust according to the structure of the graph. In the bottom-up phase, rather than expanding through a frontier, the algorithm checks each unvisited vertex whether any of its neighbours has been visited, this reduces redundant checks in densely populated regions of the graph. To assist this, the system employs data structures such as bitmaps to monitor visited vertices, statistical counters to assist in the decision to switch approaches, and efficient means of accessing adjacency lists with Hybrid Memory Cube (HMC). Such an arrangement promotes the capability of FPGA hardware to perform in parallel, seeking quick and efficient traversal.[5] Weixin Zeng et al. used knowledge graph entity alignment with a degree-aware learning and fusion approach that has three key steps, which include pre-alignment, alignment, and post-alignment. In the pre-alignment phase, there are two distinct modules

Provenance and peer-review: Unsolicited and externally peer-reviewed

Data availability statement: The Cora dataset is downloaded from https://graphsandnetworks.com/

for representation learning. First, the structural representation learning module utilizes a model such as RSNs to learn structural relationships contained in the graph. Secondly, the name representation module operates on concatenated power mean word embedding to learn and encode the textual properties of entities. The two signals explored and learned in this step are vital, especially for long-tail entities that do not have strong connections in the graph. In the alignment phase, the learned features from the structural and name representations are combined in a Degree-Aware Fusion Module, with their overall contributions dynamically weighted depending on the degree of each entity. The ability to dynamically fuse information depending on how structurally significant each entity is can enrich the alignment step; for example, if a node's has a high degree then its embedding and its neighbours are more likely to contribute to that node's final embedding in the alignment step than they are at a low degree. The post-alignment phase then takes the high confidence alignment pairs and adds them back into the knowledge graphs for iterative learning and refinement of the learned representation.[6] Combining hardware-and software-centric strategies to optimize streaming graph workloads is an alternate strategy intended to boost computational efficiency, decrease redundancy, and improve performance in real-time processing settings. The thrust of the approach is toward incremental computation models that focus on operating locality around updated vertices versus the graph itself, thus removing redundancies across a continuous stream of updates. A key innovative outcome of this work was "batch reordering" (RO); this involved changing the order of input updates to take advantage of computational locality and increase cache performance. The RO technique was especially useful to algorithms which are conventionally averse to reordering. The paper also provides an input-aware software and hardware for specific workload usages. The input-aware software and hardware was extensively evaluated on a dual-socket Intel Xeon Platinum 8180 platform, resulting in numerous workload evaluations. Performance boost evaluations found that the system speedups ranged from noted up-to 4.55x performance improvement in update throughput, on average.[7]

Classical sorting and searching algorithms form the foundation of efficient data access and traversal strategies, and their comparative performance characteristics have been extensively analyzed in the literature to understand trade-offs between time complexity and practical efficiency.[8] Several studies have proposed modified Breadth-First Search (BFS) techniques to improve traversal efficiency in directed cyclic and acyclic graphs, particularly by adapting queue management and visitation strategies to handle directionality more effectively.[9] The identification of connected components using traversal-based techniques has also been explored in the context of iterative methods for solving linear systems, highlighting the importance of traversal order in accurately detecting graph components.[10] Social network analysis provides the theoretical basis for understanding traversal behavior and node importance in complex networks, where structural properties such as degree, centrality, and connectivity directly influence algorithm efficiency and search dynamics.[11] These findings collectively motivate degree-aware traversal strategies that exploit local structural cues, which are not fully leveraged in traditional DFS-based approaches.[12]

By employing priority queues to rank nodes according to their degree, Degree-Based Search (DBS) algorithms improve graph traversal and outperform more conventional techniques like BFS and DFS, especially when it comes to locating important pathways and interconnected elements. Although DBS techniques work well in connected graphs, they have not been widely used in directed and disconnected networks, which are prevalent in real-world situations. Without extra tools to monitor unvisited nodes, traditional algorithms perform poorly in these conditions, resulting in inefficient and incomplete exploration. The suggested DBS approach incorporates degree-based selection and priority queues to overcome these drawbacks, dynamically adjusting to the characteristics of the graph. The highest-ranked unvisited node is the starting point for traversal, and nodes are ranked by total degree (sum of in-degree and out-degree). After that, neighbours are investigated recursively, giving higher out-degree neighbours priority in order to guarantee better graph coverage. This method makes it possible to thoroughly examine both sparsely connected and densely connected areas in directed graphs. This method's support for visual representation—where node size reflects out-degree and layouts aid in highlighting structural relationships—is one of its key features. The method is especially useful in fields like social network analysis, biological systems, and information retrieval because of this visual dimension, which facilitates the interpretation of node importance and network topology. This DBS algorithm prioritizes structural hierarchy and analytical clarity over hybrid FPGA-based traversals or knowledge graph alignment techniques. It improves robustness and usability in static directed transport networks and similar systems that need to identify influential nodes and links by dynamically recalculating node degrees and adapting the traversal frontier. As indicated in Table 1, the suggested Degree-Based Directed Traversal technique is contrasted with other comparable traversal methods. In contrast to other techniques that are based on switching directions or feature alignment, DBDT is based on node priority according to degree and fully supports disconnections.[13]

Methodology

We present a degree-based search algorithm that methodically investigates every component without relying on assumptions about connectivity, thus allowing effective traversal of directed disconnected graphs. The proposed degree-based traversal procedure is formally described in Algorithm 1. The prioritization mechanism is based on node degree characteristics.

Let $G = (V, E)$ be a directed graph, where $V = \{v1, v2, \ldots, vn\}$ and $E \subseteq V \times V$.

Table 1 | Comparison of degree-based traversal with related methods

Method	Traversal Focus	Disconnected Support	Domains
Hybrid Traversal (FPGA-HMC)	Direction-switching (top-down/bottom-up)	Implicit (via queues)	Hardware systems
KG Entity Alignment (RSNs)	Feature-based fusion (no traversal)	No	Knowledge graphs
Streaming Graph Computation	Region-based updates	No	Real-time systems
Degree-Based Directed Traversal (Ours)	Node priority via in/out degrees	Yes (explicit)	Citation, Social, Recommenders

For every node $v \in V$, the algorithm computes:

- In-degree: $\deg^-(v) = \{u \in V : (u, v) \in E\}$
- Out-degree: $\deg^+(v) = \{u \in V : (v, u) \in E\}$
- Total degree: $\deg(v) = \deg^-(v) + \deg^+(v)$

The Node Priority is Defined by:

$$P(v) = \alpha \cdot \deg^-(v) + \beta \cdot \deg^+(v), \text{ where } \alpha + \beta = 1 \quad (1)$$

When $\alpha > \beta$, nodes with high in-degree are prioritized (e.g., authoritative hubs), while $\beta > \alpha$ favors sources or broadcasters.

The algorithm iteratively selects the unvisited node with the highest $P(v)$ value and explores its unvisited neighbors in descending order of their priority. It maintains:

- $V_{visited} \subseteq V$, the set of visited nodes
- L_{search}, the list of nodes in the traversal order

If no neighbors are available, it restarts from the next highest-priority unvisited node, thus ensuring coverage of disconnected regions. Neighbors are ordered by $(P(u), \deg^+(u), \deg^-(u))$ lexicographically.

This approach inherently supports disconnected directed graphs, unlike BFS or DFS, which rely on reachability from a seed node. Adjusting α and β improves adaptability: e.g., α = 1 targets sinks in citation networks; β = 1 reveals spreading nodes in influence graphs.

Comparative Analysis with BFS and DFS

Breadth-First Search (BFS) explores a graph level by level. For disconnected graphs, it must be restarted per component. BFS has time complexity O(|V |+|E|) and finds shortest paths.

Depth-First Search (DFS) explores branches deeply before backtracking. It also requires reinitialization for disconnected graphs and has the same complexity O(|V | + |E|).

Degree-Based Search (DBS) differs in its global prioritization:

- Uses a scoring function *P(v)* to select significant nodes
- Traverses without needing connected components
- Prioritizes hubs or broadcasters depending on α, βDBS has a worst-case time complexity:

$$O(n^2 + m \log n)$$

Algorithm 1 | Degree-Based Traversal Algorithm

Require: Directed graph $G = (V, E)$
Ensure: Ordered list L_{search} of traversed nodes

Initialize $in\_degrees[v] \leftarrow 0$, $out\_degrees[v] \leftarrow 0$ for all $v \in V$
for each edge $(u,v) \in E$ **do**
  $in\_degrees[v] \leftarrow in\_degrees[v] + 1$
  $out\_degrees[u] \leftarrow out\_degrees[u] + 1$
end for
for each node $v \in V$ **do**
  Compute $P(v) \leftarrow \alpha \cdot in\_degrees[v] + \beta \cdot out\_degrees[v]$
where $\alpha + \beta = 1$, $\alpha, \beta \in [0, 1]$
end for
$V_{visited} \leftarrow \emptyset$, $L_{search} \leftarrow [\,]$
while $V_{visited} \neq V$ **do**
  $v_{current} \leftarrow \arg\max_{v \in V \setminus V_{visited}} P(v)$
  $V_{visited} \leftarrow V_{visited} \cup \{v_{current}\}$
  Append $v_{current}$ to L_{search}
  $N(v_{current}) \leftarrow \{u \in V \setminus V_{visited} \mid (v_{current}, u) \in E\}$
  Sort $N(v_{current})$ in descending order of:
  - $P(u)$
  - If equal, by $out\_degrees[u]$
  - If still equal, by $in\_degrees[u]$
for each $u \in N(v_{current})$ **do**
  if $u \notin V_{visited}$ **then**
    Recursively apply steps 9–13 on u
  end if
end for
end while
return L_{search}

Where:

- Degree computation: O(*m*)
- Node selection: O(n²) over n steps
- Neighbor sorting: $\sum_{v \in V} O(\deg^+(v) \log \deg^+(v)) \leq O(m \log n)$

With a priority queue optimization, the complexity can be reduced to:

$$O((|V| + |E|) \log |V|)$$

In summary, while BFS and DFS are computationally cheaper for general-purpose traversal, DBS is better suited for applications where node importance (degree centrality) and disconnected regions are key. It is particularly effective in networks like citations, social graphs, and recommender systems where structure is fragmented and influence or importance is non-uniform.

Simulation With Artificial Data

The Cora citation network, a directed and structurally fragmented graph with 2,708 papers and 5,429 citation edges spanning seven subject categories, is used to test the degree-based search algorithm. Preliminary analysis reveals 78 weakly connected components, with a dominant component encompassing approximately 88.5% of the nodes. The wide range of

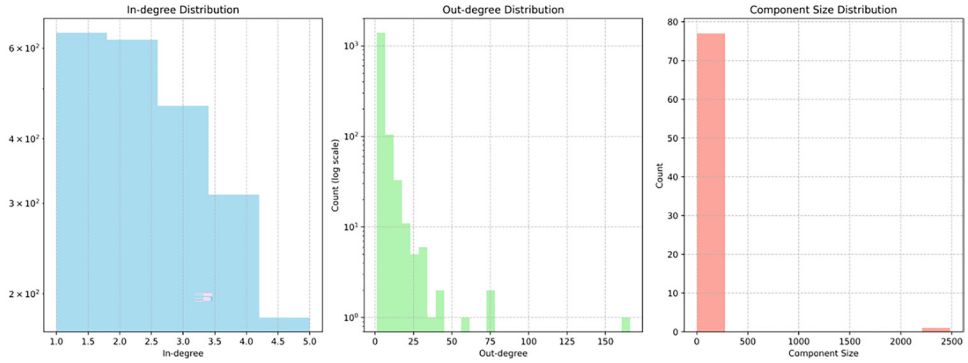

Fig 1 | The left panel shows in-degree distribution with most papers receiving 0–3 citations and peaks at 1.0 and 2.0; the center panel displays out-degree distribution on a logarithmic scale showing most papers cite fewer than 25 other works; the right panel reveals component size distribution with numerous small components and one dominant giant component containing approximately 2000+ papers

in- and out-degrees reflects the dynamics of real-world citation networks, where a small number of papers are highly cited while many remain isolated.

The proposed algorithm utilizes a tunable combination of in-degree and out-degree (denoted by parameters α and β) to assign a global priority score to each node. Unlike conventional traversal strategies that rely on initial reachability, this approach selects the highest-priority unvisited node at each iteration and recursively explores its reachable neighbors. When necessary, the algorithm restarts from the next unvisited node with the highest priority, allowing full traversal across disconnected components.

Simulation results on the Cora dataset demonstrate the algorithm's flexibility in navigating both sparsely and densely connected subgraphs. When $\beta > \alpha$, the algorithm emphasizes outward exploration by following nodes with high out-degree,

while $\alpha > \beta$ focuses on central, highly cited nodes with high in-degree. Additionally, the algorithm monitors subject label transitions, thereby capturing semantic shifts in the graph's structure.

Key performance indicators, including execution time, number of component transitions, and subject changes, are recorded across various configurations. These experiments confirm the algorithm's ability to explore intricate and fragmented citation networks while maintaining sensitivity to structural depth and semantic awareness. The results support its broader applicability to real-world scenarios such as knowledge graphs and recommender systems, where disconnectedness and topical diversity are common. Implementation details and pseudocode for DFS, BFS, and the proposed Degree-Based Search algorithm are provided in the Appendix.

Experimental Results and Analysis

Experiments were carried out in two different environments to assess the performance and reproducibility of the proposed algorithm.

A personal computer running Windows 11 (64-bit), equipped with an Intel Core i7 processor (11th Generation, 2.80GHz), 16 GB RAM, and a 512 GB SSD. The experiments were executed using Python 3.10 in a Jupyter Notebook environment. Google Colab was used for additional evaluations. The cloud environment facilitated scalable testing and allowed rapid prototyping. All algorithms were executed multiple times to account for runtime variability. Performance metrics, including traversal percentage and discovery position, were averaged over repeated runs to ensure statistical consistency. The experiment evaluates the hub node visit position in Erdős–Rényi graphs using three traversal strategies: DFS, BFS, and Degree-Based Search (DBS). The hub node, identified as the one with the highest degree, was visited first in DBS, showcasing its effectiveness in prioritizing influential nodes. In comparison, DFS and BFS visited the hub at later positions, indicating less focus on node importance. These results highlight that DBS is more efficient for early discovery of key nodes in random graph structures like Erdős–Rényi models. The citation network analysis reveals robust trends in paper impact and search efficiency. As Figure 1 shows, the Cora dataset has distributional properties in line with a power-law relationship between in-degree and out-degree distributions. The in-degree distribution shows that most papers have a small number of citations (between 0 and 5), with frequency dropping off exponentially as the number of citations increases. This is in line with the anticipated citation pattern of academic literature, where a few highly influential papers have extremely high citation frequency. The out-degree distribution is shown to display that most papers cite fewer than 25 other papers, with frequency dropping off sharply after this. Additionally, the component size distribution shows that the dataset is comprised largely of one large connected component with around over 2000 papers, with the other components being significantly smaller, showing the existence of a cohesive core research community and some peripheral research clusters.

The traversal behavior comparison in Figure 2 shows inherent differences in how each search algorithm orders papers. Classical graph traversal algorithms (DFS and BFS) traverse structure-determined paths with DFS and BFS having the same traversal order for the top

DOI: https://doi.org/10.70389/PJS.100203 | Premier Journal of Science 2025;15:100203

First 10 Papers Traversed by Each Method:

Position	DFS	BFS	DBS-0.00	DBS-0.25	DBS-0.50	DBS-0.75	DBS-1.00
1	1123188 (5 in, 0 out)	1123188 (5 in, 0 out)	35 (3 in, 166 out)	35 (3 in, 166 out)	35 (3 in, 166 out)	35 (3 in, 166 out)	294030 (5 in, 1 out)
2	545647 (3 in, 1 out)	545647 (3 in, 1 out)	887 (3 in, 27 out)	887 (3 in, 27 out)	887 (3 in, 27 out)	887 (3 in, 27 out)	20924 (3 in, 6 out)
3	582511 (3 in, 1 out)	582511 (3 in, 1 out)	6213 (3 in, 76 out)	6213 (3 in, 76 out)	6213 (3 in, 76 out)	6213 (3 in, 76 out)	521207 (5 in, 1 out)
4	118435 (0 in, 1 out)	118435 (0 in, 1 out)	6214 (4 in, 28 out)	6214 (4 in, 28 out)	6214 (4 in, 28 out)	6214 (4 in, 28 out)	521269 (2 in, 1 out)
5	118436 (4 in, 0 out)	118436 (4 in, 0 out)	6184 (4 in, 8 out)	6184 (4 in, 8 out)	6184 (4 in, 8 out)	6184 (4 in, 8 out)	1125993 (1 in, 0 out)
6	1109873 (2 in, 0 out)	1109873 (2 in, 0 out)	3229 (4 in, 61 out)	3229 (4 in, 61 out)	3229 (4 in, 61 out)	3229 (4 in, 61 out)	1108267 (5 in, 0 out)
7	294030 (5 in, 1 out)	294030 (5 in, 1 out)	35922 (4 in, 13 out)	35922 (4 in, 13 out)	35922 (4 in, 13 out)	35922 (4 in, 13 out)	1113934 (5 in, 0 out)
8	20924 (3 in, 6 out)	20924 (3 in, 6 out)	50336 (2 in, 5 out)	50336 (2 in, 5 out)	50336 (2 in, 5 out)	396412 (5 in, 0 out)	1115701 (5 in, 0 out)
9	1108267 (5 in, 0 out)	1108267 (5 in, 0 out)	50337 (3 in, 3 out)	50337 (3 in, 3 out)	50337 (3 in, 3 out)	50337 (3 in, 3 out)	289885 (4 in, 2 out)
10	1113934 (5 in, 0 out)	1113934 (5 in, 0 out)	13269 (2 in, 1 out)	13269 (2 in, 1 out)	396412 (5 in, 0 out)	390894 (3 in, 0 out)	1119708 (5 in, 0 out)

Fig 2 | The first 10 papers traversed using different search methods across a citation network, comparing traditional approaches (DFS, BFS) with Degree-Based Search (DBS) at varying α values from 0.00 to 1.00. Each row represents a position in the traversal sequence (1-10), while each column shows the paper ID followed by its in-degree and out-degree (citations received and made) for each method, revealing how increasing α values in DBS progressively prioritize papers with higher citation counts compared to the topology-driven paths of DFS and BFS

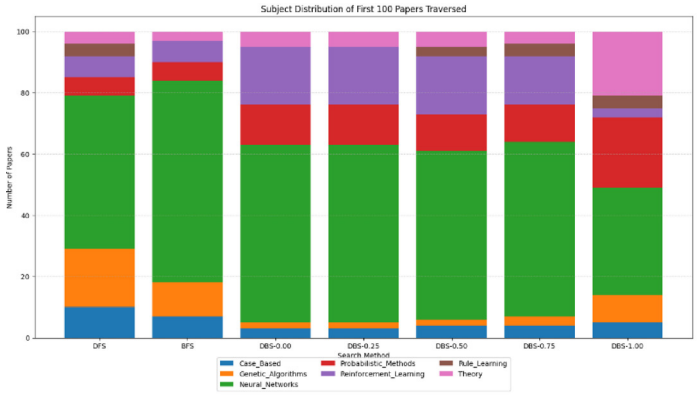

Fig 3 | This stacked bar chart illustrates the subject distribution of the first 100 papers traversed by different search methods (DFS, BFS, and DBS with various α values from 0.00 to 1.00) across a citation network. Each colored segment represents the proportion of papers from different research domains (Case-Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Rule Learning, Reinforcement Learning, and Theory), revealing how traditional methods (DFS/BFS) yield more diverse subject coverage while DBS with increasing α values shows a progressive bias toward Neural Networks (red) and Theory (pink) categories at the expense of other domains

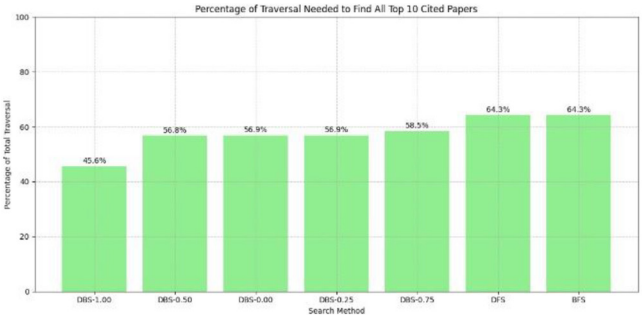

Fig 4 | Percentage of traversal required to identify all top 10 cited papers using various search strategies

10 papers. This reveals these methods to be bound by the graph topology instead of paper influence. DBS variants, however, show significantly different traversal patterns depending on their α parameter values. When α increases from 0.00 to 1.00, the algorithm increasingly favors papers with higher in-degree (more citations received). At α = 1.00, DBS instantly reconizes influential papers such as paper 293638 (5 in, 1 out) and 28924 (3 in, 6 out) in its earliest traversal steps. This illustrates DBS's efficiency in quickly finding influential papers when set to maximize citation count. The subject distribution plot in Figure 3 illustrates how every search strategy samples from throughout various research domains within the citation network. DFS is the most heterogeneously distributed, with good coverage throughout numerous different categories such as Neural Networks, Genetic Algorithms, and Rule Learning. BFS is skewed toward some research domains, such as Genetic Algorithms. The DBS variants increasingly display specialized traversal patterns with larger values of α, with DBS-1.00 providing extremely high preference toward papers in the Neural Networks domain. This illustrates that the most cited papers are clustered in specific research communities and not evenly spread throughout all subjects. The variation in subject distribution across different values of α indicates that in-degree and out-degree measures are correlated with research domain, which could be due to the differences in citation practices across various academic communities or the evolution of research focus over the length of the dataset's period.

Figure 4 displays a comparative review of the percentage of graph traversal necessary to identify the entire set of top 10 most influential papers discovered

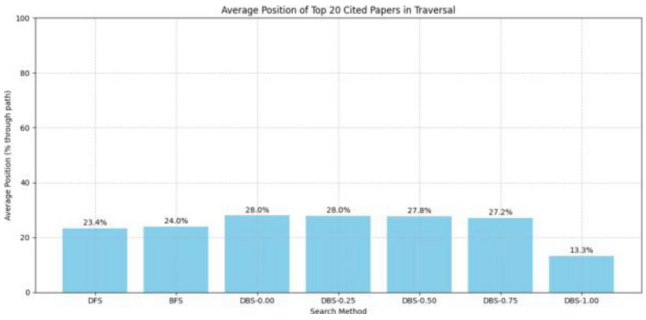

Fig 5 | Average positional rank of the top 20 most cited papers during traversal using various search strategies

by using different search strategies, including the proposed Degree-Based Search (DBS) approach utilizing different weights of the same methodology (DBS-1.00, DBS-0.75, DBS-0.50, DBS-0.25, and DBS-0.00),

Depth-First Search (DFS), and Breadth-First Search (BFS). The percentage of graph traversal can be tracked along the vertical axis as a measure of search efficiency.

The most efficient traversal of all methods was demonstrated by the DBS-1.00 search, requiring a total of 45.6% of the graph to be traversed to identify the set of top 10 papers. This represents a significant improvement over an amount of traversal of 64.3% in either the DFS or BFS method for the same number of papers, which also suggests a clear benefit to ranking papers in descending order of degree.

The performance of the additional DBS variants (with weights varying from 0.00 to 0.75) is also relatively consistent, with traversal proportions ranging from 56.3% to 58.5%. These findings suggest that while some degree-based prioritization attains efficiency, maximal benefit is gained when the algorithm strictly utilizes degree in descending order (DBS-1.00).

Together, the study shows that discovering highly cited nodes can be dramatically accelerated using a degree-based search strategy, which should prove to be an effective, scalable alternative for citation network analysis and comparable graph-based applications.

Figure 5 portrays the average positional rank of the top 20 most cited papers identified during traversal, expressed as a percentage of the total traversal length. This measure provides insight into how early high-impact nodes are discovered based on differences in exploration strategy.

The DBS-1.00 variant demonstrates the most efficacious identification of highly cited nodes, occurring on average at just 13.3% of the traversal, outperforming all other methods. In contrast, traditional strategies such as Depth-First Search (DFS) and Breadth-First Search (BFS) required exploring a larger share of the graph to locate the same influential papers, with average positions of 23.4% and 24.0%, respectively.

This substantial difference indicates that strict degree-based ordering (DBS-1.00) not only reduces the total traversal cost but also significantly improves the timeliness of discovering key nodes. Other DBS variants (DBS-0.00 to DBS-0.75) show modest performance gains, with average discovery positions ranging between 26.3% and 27.8%.

These results reinforce earlier findings: the benefit of degree-based strategies is most prominent when node traversal is prioritized strictly by descending degree. The ability to identify such nodes early is particularly important in research domains such as citation analysis, influence maximization, and recommendation systems, where timely recognition of key elements can enhance both performance and decision-making.

Overall, the evidence supports the conclusion that the DBS-1.00 approach offers both efficient exploration and rapid access to high-value nodes, positioning it as a highly effective and scalable algorithm for graph-based research applications.

Conclusions

This work presented a Degree-Based Search algorithm that uses a priority function based on a tuneable combination of in-degree and out-degree to navigate disconnected directed graphs. By avoiding the need for root-based reachability, the technique makes it possible to systematically investigate fragmented topologies. Experiments on synthetic Erdős–Rényi graphs and the Cora citation dataset showed that the algorithm is able to adapt well to structural diversity, spanning multiple components through dynamic re-initialization and identifying influential nodes. However, the degree distribution of the input graph is inherently linked to the method's performance. In networks without degree heterogeneity, where prioritization provides little benefit, its efficacy decreases. Furthermore, further tests on other disconnected networks in the real world are required to confirm generalizability, even though the Cora dataset offered a realistic benchmark. Additionally, the current implementation is sequential; parallelization could result in additional performance improvements.

Future research will examine hybrid strategies that combine DBS with conventional search techniques (such as BFS, shortest path heuristics), as well as graph embeddings, in order to overcome these constraints. These combinations might provide better coverage and efficiency, particularly in graphs with low-degree homogeneity or subtle structure. There is also potential for applications in evolving and sparse information

DOI: https://doi.org/10.70389/PJS.100203 | Premier Journal of Science 2025;15:100203

when the framework is extended to probabilistic traversal or reinforcement learning settings.

References

1 Kaviyarasu M, Aslam M, Afzal F, et al. The connectivity indices concept of neutrosophic graph and their application of computer network, highway system and transport network flow. Sci Rep. 2024;14:4891. https://doi.org/10.1038/s41598-024-54104-x

2 Tripathy A, Panada AC, Behera SP, Mohanty BS. Edge connectivity of a neutrosophic graph. Neutrosophic Sets Syst. 2025;81:1.

3 Zhong Z, et al. Connectivity determination algorithm for complex directed networks. IEEE Trans Netw Sci Eng. 2025. https://doi.org/10.1109/TNSE.2025.3549777

4 Zhang J, Li J. Degree-aware hybrid graph traversal on FPGA-HMC platform. In: Proc ACM/SIGDA Int Symp Field Program Gate Arrays; 2018. p. 229–38. https://doi.org/10.1145/3174243.3174245

5 Zhang J, Li J. Degree-aware hybrid graph traversal on FPGA-HMC platform. In: Proc 2018 ACM/SIGDA Int Symp Field-Programmable Gate Arrays (FPGA '18); 2018. p. 229–38. https://doi.org/10.1145/3174243.3174245

6 Zeng W, Zhao X, Wang W, Tang J, Tan Z. Degree-aware alignment for entities in tail. In: Proc 43rd Int ACM SIGIR Conf Res Dev Inf Retr (SIGIR '20); 2020. p. 811–20. https://doi.org/10.1145/3397271.3401161

7 Basak A, Qu Z, Lin J, Alameldeen AR, Chishti Z, Ding Y, Xie Y. Improving streaming graph processing performance using input knowledge. In: Proc MICRO-54: 54th Annu IEEE/ACM Int Symp Microarchit (MICRO '21); 2021. p. 1036–50. https://doi.org/10.1145/3466752.3480096

8 Subbarayudu B, Gayatri LL, Nidhi PS, Ramesh P, Reddy RG, Reddy CKK. Comparative analysis on sorting and searching algorithms. Int J Civ Eng Technol. 2017.

9 Baidari I, Hanagawadimath A. Traversing directed cyclic and acyclic graphs using modified BFS algorithm. In: Proc 2014 Sci Inf Conf (SAI); 2014. p. 175–81. https://doi.org/10.1109/SAI.2014.6918187

10 Prolubnikov A. Finding connected components of a graph using traversals associated with iterative methods for solving systems of linear equations. arXiv. 2024;2407.10790. Available from: https://doi.org/10.48550/arXiv.2407.10790

11 Tabassum S, Pereira F, Fernandes S, Gama J. Social network analysis: an overview. Wiley Interdiscip Rev Data Min Knowl Discov. 2018;8(5):e1256. https://doi.org/10.1002/widm.1256

12 Allender E, Chauhan A, Datta S. Depth-first search in directed planar graphs, revisited. Acta Inform. 2022;59. https://doi.org/10.1007/s00236-022-00425

13 Shyma PV, Shanker KPS. Degree based search: a novel graph traversal algorithm using degree based priority queues. Int J Adv Comput Sci Appl. 2024;15:1366–71.

Appendix

Graph Traversal Algorithms in Python

Depth-First Search (DFS) Traversal

```
def dfs_traversal(G, start_node=None):
  """Perform a standard DFS traversal on the graph"""
    visited = set()
    path = []
    def dfs(node):
        if node in visited:
            return
      visited.add(node)
      path.append(node)
      for neighbor in G.successors(node):
            if neighbor not in visited:
            dfs(neighbor)
# Start from node with highest in-degree if not specified
if start_node is None:
    start_node = max(dict(G.in_degree()). items(), key=lambda x: x[1])[0]
dfs(start_node)
# Continue until all nodes are visited
while len(visited) < G.number_of_nodes():
    unvisited = list(set(G.nodes()) -visited)
    dfs(unvisited[0])
return path
```

Listing 1: DFS Traversal on a Directed Graph

Breadth-First Search (BFS) Traversal

```
from collections import deque
def bfs_traversal(G, start_node=None):
    """Perform a standard BFS traversal on the graph"""
    visited = set()
    path = []
    def bfs(start):
     queue = deque([start])
visited.add(start)
 while queue:
    node = queue.popleft()
    path.append(node)
    for neighbor in G.successors(node):
      if neighbor not in visited:
         visited.add(neighbor)
          queue.append(neighbor)
if start_node is None:
    start_node = max(dict(G.in_degree()).items(), key=lambda x: x[1])[0]
bfs(start_node)
    while len(visited) < G.number_of_nodes():
unvisited = list(set(G.nodes()) -visited)
bfs(unvisited[0])
    return path
```

Listing 2: BFS Traversal on a Directed Graph

Degree-Based Search (DBS) Traversal

```
def degree_based_search(G, weak_components, alpha=0.5, beta=0.5):
    """Perform a degree-based search traversal of the graph"""
      in_deg = dict(G.in_degree())
      out_deg = dict(G.out_degree())
    score = {v: alpha * in_deg[v] + beta * out_deg[v] for v in G.nodes()}
      visited = set()
      path = []
    node_to_comp = {node: i for i, comp in enumerate(weak_components) for node in comp}
      comp_trans = 0
      prev_comp = None
      def dfs(node):
    nonlocal comp_trans, prev_comp
    if node in visited:
       return
    current_comp = node_to_comp[node]
    if prev_comp is not None and prev_comp!= current_comp:
      comp_trans += 1
    prev_comp = current_comp
    visited.add(node)
    path.append(node)
neighbors = [n for n in G.successors(node) if n not in visited]
neighbors.sort(key=lambda x: (score[x], out_deg[x], in_deg[x]), reverse=True)
    for n in neighbors:
      dfs(n)
    while len(visited) < G.number_of_nodes():
     unvisited = list(set(G.nodes()) -visited)
         current = max(unvisited, key=lambda v: score[v])
      dfs(current)
    return path, comp_trans, score
```

Listing 3: Degree-Based Search (DBS) with α and β Parameters

DOI: https://doi.org/10.70389/PJS.100203 | Premier Journal of Science 2025;15:100203