Algorithmic Detection and Comparative Analysis of Critical Points in Real-World Infrastructure Networks

Somda's Team: Somendra Singh Kharola; Sasapu Lokeswar Rao. Team ID: T22

Contents

0.1 Delineation of Project Objectives

This section provides a comprehensive overview of the tasks and experiments conducted in our project. The successful completion of the project's primary and secondary objectives is detailed across various sections.

Primary Objective: Task Completion The three primary tasks for this project have been successfully completed and are detailed in the following sections of the report:

- **Task 1:** Implement algorithms to detect critical edges and vertices in graphs.

  This is delineated in Section 5 (Time and Space Complexity Analysis), Section 7 (Code Implementation Overview), and Section 8 (Results and Discussion).

- **Task 2:** Validate results on test cases with known bridges and articulation points.

  This is delineated in Section 7 (Description of Experiments) and Section 8 (Correctness and Validation).

- **Task 3:** Discuss applications in network reliability and clustering.

  This is delineated in Section 9 (Applications of Critical-Points-Detecting-Algorithms to Network Reliability).

Primary Objective: Experiment Completion The three primary experiments for this project have been successfully completed, with their methodologies and results detailed in the report:

- **Experiment 1:** Test on known benchmark graphs with documented critical points.

  This is delineated in Section 7 (Description of Experiments and Datasets Used) and Section 8 (Correctness and Validation).

- **Experiment 2:** Analyze sensitivity to graph density.

  This is delineated in Section 8.2 (Analysis of Sensitivity to Graph Density).

- **Experiment 3:** Measure runtime with increasing graph complexity.

  This is delineated in Section 8.3 (Runtime with Increasing Graph Complexity) and Section 8.4 (Network Analysis Robustness and Analysis).

Secondary Objective: Sub-Project Completion The successful completion of the secondary objective, the sub-project titled **"Finding Crucial Connections in a Power Grid Whose Failure Would Split the System,"** is delineated in its entirety in **Section 3.1** (Sub-Project: Finding Critical Points in Power Grid Whose Failure Would Split the System). This section provides an in-depth, quantitative analysis of the `power_grid_uci.txt` network, including multi-point failure simulations and a detailed risk assessment.

1 Declarations and Acknowledgements

Please note: The source code of the secondary objective – a sub-project titled **"Finding Crucial Connections in a Power Grid Whose Failure Would Split the System"** is independent of the source code of the primary objective, because we exploited the extensive libraries of Python, after learning them from **Mr. Aditya Garg, M.Math, First Year, ISI, Bangalore**, to whom we are eternally grateful.

Mr. Aditya Garg also helped us streamline our workflow and coding to make it what it is now. Lastly, to understand the working for certain very specific libraries and functions and as a thesaurus to find academically respectful technical terms and definitions (in certain very niched areas of computer science), **AIs bots Chat GPT, Gemini and DeepSeek were used**.

Lastly, we are grateful to **Prof. Jaya Sreevalsan Nair**, **IIIT Bangalore**, and the instructor of this course – Design and Analysis of Algorithms – being conducted in fall semester of 2025, ISI Bangalore, to have given us this opportunity to learn certain core principles of algorithmic design through a project that uses real-world data.

2 Introduction

Transportation systems, power grids, and communication networks are all examples of infrastructure networks that modern society needs to work. These systems are made up of many parts that are all connected in complicated ways.

So, it's not surprising that if even one part fails, it can cause a chain reaction of problems that have big effects on society and the economy. The examination of network resilience—the capacity of a network to endure such failures—is consequently an essential domain within engineering and computer science.

Graph theory offers a robust and universal framework for modeling these systems, representing components as vertices (V) and their connections as edges (E) in a graph G = (V,E). This model allows for precise quantification of structural vulnerability. Articulation points (or cut vertices) and bridges (or cut edges) are the most basic signs of fragility.

Simply put, An articulation point in graph theory is a vertex whose removal increases the number of connected components in a graph. A bridge is an edge that, when taken away, breaks the network. These parts are single points of failure; if they fail, the only way for parts of the network to connect is broken, which causes fragmentation. How and why a network is built has a big effect on how common these critical points are.

Indeed, geography and cost often limit physical infrastructures, such as roads and power grids. This results in hierarchical or tree-like graphs that have many weaknesses. Internet networks, on the other hand, are built to be more reliable by having many backup connections (redundancies), which gives them a mesh-like shape.

2.1 Fundamental Differences Between Articulation Points and Bridges

While both articulation points and bridges represent single points of failure, their structural roles and the potential impact of their removal are fundamentally different. The distinction is critical for network reliability engineering and vulnerability assessment.

2.1.1 Impact on Network Fragmentation

Indeed, the most important difference is the size and predictability of the damage that happens when they fail. Our empirical findings distinctly illustrate this differentiation, (see for instance Section 3.1).

- **A Bridge has a predictable, binary impact.** When a bridge is removed, it is guaranteed to increase the number of connected components by exactly one. For a connected graph, this means it will always split into precisely two pieces. As seen in the analysis of the `power_grid_uci.txt` dataset, every one of the 2,736 bridge removals resulted in exactly 2 fragments.
- **An Articulation Point can have a catastrophic, multi-fragment impact.** Removing an articulation point can split a graph into two, three, or potentially hundreds of components. It acts as a central hub connecting multiple, otherwise separate, regions. The analysis of `roadNet-PA.txt`, where removing a single AP (node 689085) created 207 components, is a perfect real-world example of this devastating potential.

2.1.2 Formal definitions of an Articulation Point (AP) and Bridge

Let $G = (V, E)$ be a graph and $k(G)$ be the number of its connected components. A vertex $v \in V$ is an articulation point if the subgraph $G - v$ (formed by removing v and all incident edges) satisfies the condition:

$$k(G - v) > k(G)$$

Equivalently, a vertex v is an articulation point if there exist two other distinct vertices u, w such that every path between u and w contains v.

Let $G = (V, E)$ be a graph and $k(G)$ be the number of its connected components. An edge $e \in E$ is a bridge if the subgraph $G - e$ (formed by removing the edge e) satisfies the condition:

$$k(G - e) > k(G)$$

Equivalently, an edge e is a bridge if and only if it does not belong to any cycle in G.

2.1.3 Difference in Structural Role

The distinct impact of each critical point stems from its unique role within the network's topology.

- **A Bridge is defined by a lack of redundancy.** It is an edge that is not part of any cycle. If one is at an endpoint of the bridge, the only way to reach the other endpoint via that specific path is to cross it; there are no "detours" that involve other edges to form a loop.
- **An Articulation Point is defined by its necessity for paths between *other* nodes.** It serves as a gateway. Two distinct nodes, u and w, might have multiple paths between them, but if all of those paths require passing through a third node, v, then v is an articulation point.

The relationship between articulation points and bridges exhibits important mathematical properties:

- If edge (u, v) is a bridge and both endpoints have degree ≥ 2, then neither u nor v is necessarily an articulation point
- If edge (u, v) is a bridge and either endpoint has degree 1, then the vertex with degree ≥ 2 is an articulation point
- A graph can contain bridges without articulation points (e.g., two cycles connected by a single bridge edge)
- The presence of articulation points often indicates more severe structural vulnerabilities than bridges alone

2.1.4 Computational Complexity

Both articulation points and bridges can be detected efficiently using depth-first search (DFS) based algorithms, notably Tarjan's algorithm, which operates in $O(|V| + |E|)$ time complexity. This linear-time complexity makes these critical elements computationally tractable for large-scale network analysis, infrastructure assessment, and reliability engineering applications.

This will be discussed in greater detail, as instructed by the guildelines of the project, in a latter section, in particular, Section 5 of this report.

3 Problem Statement and Motivation

The primary objective of this project is to use an algorithm(s) to find bridges and articulation points in six large-scale, real-world networks or datasets. This will show how the designs of the six networks are different in some important ways (through articulation points and bridges). Specifically, **the primary objective** comprises three experiments and three tasks.

The three tasks are as follows:

Task 1: Implement algorithms to detect critical edges and vertices in graphs.

Task 2: Validate results on test cases with known bridges and articulation points.

Task 3: Discuss applications in network reliability and clustering.

The three experiments are as follows:

Experiment 1: Test on known benchmark graphs with documented critical points.

Experiment 2: Analyze sensitivity to graph density.

Experiment 3: Measure runtime with increasing graph complexity.

The secondary objective of this project is to "Find critical connections in a given power grid dataset whose failure would split the grid system".

While the secondary objective is closely related to the 3 experiments and 3 tasks of the primary objective, but delves deeper into the given dataset while analysing other nuances of critical points and their fragmentation impacts.

To conclude, this research project follows a systematic three-part methodology to analyze network resilience through graph theory, namely the impact of removal of APs and bridges:

- **Implementation:** To develop a robust and scalable Python implementation of the canonical depth-first search (DFS)-based algorithm for identifying all bridges and articulation points in a graph.
- **Validation:** To rigorously verify the algorithm's correctness by submitting it to a battery of synthetic and benchmark test graphs.
- **Application and Analysis:** To apply the algorithm, after successful completion of testing and validation, to large-scale datasets representing Road Networks, Internet Networks, and Electrical Power Grids and analyse the data so generated via the lenses of various quantitative metrics, each of which will be discussed in the later sections.

3.1 Sub-Project: Finding Critical Points in a Power Grid Whose Failure Would Split the System

3.1.1 Experimental Framework and Methodology

This analysis gives a full structural vulnerability assessment of the Power Grid UCI network (`power_grid_uci.txt`), which has 6,659 nodes and 8,309 edges. The study used graph-theoretic methods to thoroughly test how resilient the network was. The main goal is to go beyond simple statistical summaries and figure out exactly how single-point and multi-point failures affect the network, with a focus on network fragmentation and keeping the network's core intact, as measured by the Largest Connected Component (LCC).

The analysis was done with a Python script that used the NetworkX library and had three steps that happened one after the other:

1. **Articulation Point (AP) Analysis:** In a series of separate simulations, every articulation point (critical node) that had been found was removed. The code kept track of how many network fragments and how big the LCC were after each removal.
2. **Bridge Analysis:** Bridge Analysis: Similarly, each identified bridge (critical edge) was taken out in separate simulations to see how it affected fragmentation and LCC size.
3. **Multi-Point Failure Simulation:** To model more complex failure scenarios, 100 trials were conducted where three randomly selected APs were removed simultaneously, with the resulting fragmentation recorded for each trial.

This multi-faceted approach helps us understand the network's main failure modes and vulnerability profile in detail.

3.1.2 Comprehensive Vulnerability Assessment

The analysis script was run on the dataset, which at first had only one component.

Articulation Point (Node) Analysis: There were 1,655 articulation points found in all. The complete simulation of removing each AP gave an objective/numerical understanding of how fragile the network was.

Table 1: Articulation Point Classification Summary

Classification	Count	Percentage of All APs
Critical Hubs (>49 fragments)	0	0.00%
High-Impact APs (10-49 fragments)	1	0.06%
Moderate-Impact APs (2-9 fragments)	645	39.0%
Low-Impact APs (1 fragment)	1,009	61.0%

The analysis discovers that Node **3088** is the only high-impact vulnerability, and getting rid of it creates 13 separate parts. Still, even in this worst-case scenario of a single-node failure, the network shows commendable LCC retention.

Table 2: Impact Analysis of Top 5 Most Critical Nodes

Node ID	Fragments Created	LCC Size	LCC Retention (%)
3088	13	6,633	99.62%
1709	9	6,650	99.88%
2117	9	6,649	99.86%
735	8	6,645	99.80%
1415	8	6,646	99.82%

The LCC analysis shows an important point: even though the number of components is very low, the network still keeps its connections very well. After removing one AP, the average size of the LCC stays at **99.96%** of the other nodes. This number shows that single-node failures almost always only disconnect very small, peripheral subgraphs (often single nodes), while the main part of the network stays connected.

Analysis of Network Resilience through Bridge Removal Impact: The observed constancy in Largest Connected Component (LCC) size across multiple bridge removals reveals fundamental structural properties of the network under examination. As demonstrated in Table 1, the removal of any of the ten most critical bridges results in minimal disruption to network connectivity, with LCC sizes consistently maintaining values between 6,655 and 6,658 nodes, representing 99.94% to 99.98% of the original network connectivity.

Table 3: Top 10 Most Critical Bridges by Fragments Created

Bridge	Fragments	Increase	LCC Size	LCC %
(1, 5089)	2	1	6,658	99.98%
(2852, 1200)	2	1	6,658	99.98%
(2852, 1201)	2	1	6,658	99.98%
(761, 5078)	2	1	6,656	99.95%
(3051, 4071)	2	1	6,655	99.94%
(5, 4599)	2	1	6,658	99.98%
(5, 5100)	2	1	6,658	99.98%
(4065, 948)	2	1	6,658	99.98%
(8, 4067)	2	1	6,658	99.98%
(4067, 2978)	2	1	6,658	99.98%

This remarkable consistency in LCC preservation indicates a network architecture characterized by significant structural redundancy and robust connectivity patterns. The mathematical implication is straightforward yet profound: each bridge removal isolates only 1-4 nodes from the main component, as evidenced by the minimal reduction from the theoretical maximum of 6,659 nodes. This behavior can be expressed as:

$$\Delta_{\text{LCC}} = N_{\text{total}} - \text{LCC}_{\text{post-removal}} \leq 4$$

where $N_{\text{total}} = 6,659$ represents the complete node set.

The near-invariant LCC size suggests a **core-periphery structure** wherein bridges primarily connect peripheral nodes or small subgraphs to a densely interconnected central core. This structural configuration ensures that single-point failures have localized rather than systemic consequences. The network demonstrates engineered resilience through redundant pathways that maintain connectivity despite the failure of individual critical links.

From a network robustness perspective, this pattern contrasts sharply with more fragile architectures where bridge removal could cause catastrophic fragmentation. In such vulnerable networks, LCC size might decrease substantially, potentially splitting the network into large, disconnected components. The observed behavior here aligns more closely with robust infrastructure networks

like internet backbones, where local failures can be contained without disrupting global connectivity.

The practical significance of this finding lies in the network's ability to maintain operational integrity under stress conditions. With LCC preservation exceeding 99.9% for all critical bridge removals, the network demonstrates exceptional fault tolerance. This characteristic is particularly valuable in critical infrastructure applications where service continuity is paramount, as localized failures do not cascade into widespread service disruption.

Multi-Point Failure Escalation The coordinated failure analysis shows a measurable synergistic effect on fragmentation, which points out a major weakness that single-point tests don't show.

Table 4: Multi-Point Failure Impact Analysis (3 APs)

Metric	**Value**
Average Fragments Created	6.25
Maximum Fragments Observed	20
Minimum Fragments Observed	4

The multi-point failure simulation shows that coordinated attacks can have effects that are much worse than individual failures. The most fragmentation seen (20 parts, up from 13) is a **53.8% increase** over the most damage from the worst single-point failure (13 parts). This result implicitly demonstrates the inter-dependncies of articulation points.

3.1.3 Security Implications and Risk Assessment

Primary Vulnerabilities and Attack Vectors We can quantitatively assess three primary attack vectors based on the data:

1. **Targeted Single-Point Attack:** An attacker aiming for maximum damage from a single strike would target AP **3088**. However, this is a evidently (given the AP/bridge impact metrics) a largely inefficient attack. It disconnects only 25 nodes (0.38% of the network) while leaving 99.62% of the network operational in the LCC. The network is therefore highly resilient to single targeted attacks aimed at maximizing fragmentation.
2. **Random or Non-Targeted Failure:** A random failure of a single critical component has a high probability of being a low-impact event. Of the 4,391 identified critical points (1,655 APs + 2,736 bridges), 3,745 of them (1,009 Low-Impact APs + 2,736 Low-Impact Bridges) or **85.3%**, fall into the lowest impact category. This confirms that random outages will overwhelmingly be localized and small.
3. **Coordinated or Cascading Failure:** This is the most formidable threat. The multi-point simulation shows that removing just three APs can cause up to 20 fragments, potentially even more given we had a trial run of only 100 repeats of three-point-attacks. A planned physical attack on a few well-chosen substations (APs) or natural calamity/act-of-God event or event that causes a cascading failure in a small area could cause a disruption that is more significant than any one failure.

Resilience Characteristics Despite its vulnerabilities, the network possesses strong resilience characteristics:

- **Strong LCC Preservation**: The network maintains exceptional connectivity (LCC $> 99.6\%$) under all single-point failure scenarios, ensuring bulk connectivity remains operational.
- **Localized Impact and Gradual Degradation**: Most failures cause minimal disruption to the core, and the network's performance declines progressively rather than catastrophically.
- **Hierarchical Fragmentation Pattern:** The fact that small components of the graph tend to detach off the main core of the network and creates natural isolation boundaries that could stop cascading failures from spreading.

3.1.4 Conclusion and Recommendations

The analysis quantitatively determines that the principal vulnerability of the `power_grid_uci` network is not the presence of catastrophic single points of failure, but rather its **pervasive, systemic fragility exacerbated by multi-point failures**, specifically a considerable number of access points and bridges. Its structure makes it weak to cascading or coordinated events, even though it is very resistant to single-targeted attacks. The main security problem is that it is hard to protect the 1,655 distributed articulation points and stop multi-point attacks from snow-balling.

To further enhance and improve the analysis and depth of this project, the following pathways for further investigation are suggested (something we could not delve into owing to time and processing power constraints):

- **Flow and Capacity Integration:** This graph-theoretic analysis assumes all nodes are equal. A more advanced study would integrate data on power flow or node capacity. A "Low-Impact" AP that is actually a high-capacity substation would be functionally far more critical than its structural analysis suggests.
- **Advanced Multi-Point Simulation:** Instead of random triplets, simulate the targeted removal of the top-N most critical APs (e.g., nodes 3088, 1709, and 2117) to model a worst-case coordinated attack. Furthermore, simulate the removal of geographically proximate APs to model localized disasters.
- **Component Size Distribution:** Beyond tracking only the LCC, analyze the full distribution of fragment sizes. Understanding if failures create a few large components or many single-node "dust" components (as is the case here) provides deeper insight into the fragmentation pattern.

By integrating these additional layers of data and simulation, a more complete and actionable model of the network's resilience may be developed.

4 Description of the Approach and Method Used

Now, we shall delve into the primary objective of the project, wherein we complete the three tasks and three experiments as outlined in the excel sheet shared with our class: "daa-assignment-project-list.xls.

The methodology for the primary objective of this project **(as stated in the introduction, we made use of the "networkx" library of Python for our secondary objective (discussed just before this section) of this project)** is centered on the implementation of the well-established, DFS-based algorithm for finding bridges and articulation points, first formalized by Robert Tarjan. The algorithm performs a single pass of Depth-First Search over the graph $G = (V, E)$, maintaining two key values for each vertex u:

- **Discovery Time (`disc[u]`):** A timestamp indicating the order in which vertices are visited.
- **Low-Link Value (`low[u]`):** The lowest discovery time reachable from u by traversing zero or more tree edges and at most one back-edge.

The critical points are identified using the computed `disc` and `low` values:

- An edge (u, v) is a bridge if and only if $\texttt{low}[v] > \texttt{disc}[u]$.
- A vertex u is an articulation point if it has a child v such that $\texttt{low}[v] \geq \texttt{disc}[u]$ (with a special case for the root).

To handle the scale of large network datasets, Python's recursion limit was increased to a safe margin (`sys.setrecursionlimit(2000000)`). The graph is represented using `defaultdict(list)` for an efficient adjacency list.

5 Time and Space Complexity Analysis of Tarjan's Algorithm

5.1 Preliminaries

Let the graph be denoted by $G = (V, E)$, where $|V| = n$ is the number of vertices and $|E| = m$ is the number of edges. The implementation represents the graph as an adjacency list using a `defaultdict(list)` structure, where each edge (u, v) is stored twice (once for each direction).

5.2 Algorithmic Framework and Core Data Structures

The algorithm employs an adjacency list representation for graph $G = (V, E)$, where $|V| = n$ vertices and $|E| = m$ edges. The primary data structures are defined as follows:

- **g:** A `defaultdict(list)` representing the adjacency list, where $\forall u \in V, g[u] = \{v \mid (u, v) \in E\}$.
- **visited:** A boolean array of size n tracking visited vertices during traversal.
- **disc:** An integer array mapping vertices to discovery times during DFS.
- **low:** An integer array storing low-link values for each vertex.
- **parent:** An integer array maintaining parent-child relationships in the DFS tree.
- **bridges:** A list accumulating identified bridge edges.
- **aps_flags:** A boolean array marking articulation points.

The algorithm implements a modified Depth-First Search with the following formal specification:

For each vertex u during DFS traversal:

- Set `visited[u]` $\leftarrow$ True, `disc[u]` $\leftarrow t$, `low[u]` $\leftarrow t$, then increment t.
- For each neighbor $v \in g[u]$:
  - If v is unvisited: recursively process v, then update `low[u]` $\leftarrow \min(\texttt{low}[u], \texttt{low}[v])$.
  - If $parent[u] \neq -1 \wedge \texttt{low}[v] \geq \texttt{disc}[u]$: set `aps_flags[u]` $\leftarrow$ True.
  - If $\texttt{low}[v] > \texttt{disc}[u]$: add (u, v) to `bridges`.
  - If v is visited and not the parent: update `low[u]` $\leftarrow \min(\texttt{low}[u], \texttt{disc}[v])$.
- If u is the root and has more than 1 child: set `aps_flags[u]` $\leftarrow$ True.

5.3 Time Complexity Analysis

5.3.1 Building the Adjacency List

Each call to the `add(a, b)` method inserts both b into the adjacency list of a and a into the adjacency list of b. Since each undirected edge results in two insertions, the construction of the adjacency list requires $O(m)$ time.

5.3.2 Depth-First Search Traversal

The main computational work is performed by the recursive `dfs()` function. The DFS visits each vertex once and explores each edge exactly twice (once from each endpoint).

- **Visiting a vertex:** Assigning discovery and low values, marking a vertex as visited, and incrementing the timestamp are constant-time operations. Across all vertices, this contributes $O(n)$ time.
- **Exploring adjacency lists:** For every vertex u, the algorithm iterates through each neighbor $v \in g[u]$. Each edge is considered twice (once from u and once from v), resulting in a total contribution of $O(m)$.

– **Bridge and articulation checks:** For each edge, the algorithm performs constant-time comparisons. These checks contribute $O(1)$ per edge, for an overall contribution of $O(m)$.

Hence, the total running time for the DFS phase is $T_{\text{DFS}} = O(n + m)$.

5.3.3 Post-Processing

After traversal, the algorithm scans through the `aps_flags` array to collect articulation points. This takes $O(n)$ time.

5.3.4 Counting Connected Components

For verification of the removal effects, the algorithm calls `count_components_after_removal()` and `count_components_after_bridge_removal()`, which perform a BFS/DFS over all vertices and edges, also in $O(n + m)$ time. Since this step is repeated a constant number of times, it does not affect the asymptotic bound.

5.3.5 Total Time Complexity

Combining the above results: $T(n, m) = O(n + m)$. This bound is optimal since every vertex and edge must be examined at least once to determine connectivity properties in a general graph.

5.4 Space Complexity Analysis

The space complexity is determined by the storage required for the graph representation and data structures.

Table 5: Space complexity breakdown of data structures

Structure	Description	Space Usage
g	Adjacency list (2 entries per edge)	$O(n + m)$
visited	Boolean visited array	$O(n)$
disc	Discovery time array	$O(n)$
low	Low-link value array	$O(n)$
parent	Parent tracking array	$O(n)$
bridges	List of identified bridges	$O(m_b) \leq O(m)$
aps_flags	Boolean array marking articulation points	$O(n)$
Recursion stack	Maximum depth can be up to n	$O(n)$

Summing these contributions gives the overall space complexity: $S(n, m) = O(n + m)$**.**

5.5 Additional Considerations

5.5.1 Recursion Depth

The recursion depth in the DFS procedure is proportional to the longest path in the DFS tree, which in the worst case (e.g., a linear chain) is $O(n)$. Hence, the algorithm requires sufficient recursion limit settings, as reflected by the statement `sys.setrecursionlimit(2000000)` in the implementation.

5.5.2 Constant Factors and Practical Performance

While the theoretical complexity is linear, the practical runtime constants depend on array accesses and Python's interpreter details. However, since these are $O(1)$, the algorithm scales linearly with the number of edges and vertices. For instance, on the `as20000102.txt` dataset with $n = 6474$ and $m = 13895$, the observed runtime is below one second, consistent with the expected $O(n + m)$ growth.

Table 6: Time and space complexity by phase

Phase	Description	Time	Space
Adjacency List Construction	Builds undirected structure	$O(m)$	$O(n+m)$
DFS Traversal	Computes discovery and low values	$O(n+m)$	$O(n+m)$
Result Aggregation	Collects critical points	$O(n)$	$O(n)$
Component Counting	Used for verification only	$O(n+m)$	$O(n+m)$

5.6 Conclusion on Algorithmic Complexity

The application of Tarjan's algorithm realizes the theoretically ideal complexity limits for both temporal and spatial dimensions. The method works in linear time with respect to the number of vertices and edges, which means that $T(n,m) = O(n+m)$ and $S(n,m) = O(n+m)$. It also needs recursion depth that is proportional to $O(n)$. The analysis shows that the algorithm works well for large, undirected graphs.

6 Code Implementation Overview

The main part of the Python implementation is a Graph class that includes all the features needed to find articulation points and bridges using Tarjan's algorithm.

6.1 Core Algorithm Implementation

The algorithm works by using a modified Depth-First Search (DFS) traversal that keeps:

- **Tracking Discovery Time:** Each vertex has a timestamp that shows when it was found.
- **Calculating the Low-Link Value:** The algorithm finds the earliest accessible vertex in the DFS tree for each vertex.
- **Parent-Child Relationship Tracking:** Keeps the DFS tree structure intact.

6.2 Key Implementation Features

- **Recursion Management:** To deal with large network datasets, the recursion limit is increased.
- **Efficient Graph Representation:** Uses `defaultdict(list)` for $O(1)$ adjacency lookups.
- **Node Index Mapping:** Changes random node identifiers into sequential indices so that array-based operations can be done quickly.
- **Comprehensive Validation:** Includes ways to check important points by removing them and measuring how they affect connectivity, i.e. their impact on fragmentation.

6.3 Algorithm Workflow

The implementation follows this precise workflow:

1. **Graph Construction:** Build an adjacency list from the input data.
2. **DFS Initialization:** Initialize discovery time, low-link values, and parent arrays.
3. **Recursive DFS Traversal:** Process each vertex, updating low-link values and checking for bridge/AP conditions.
4. **Critical Point Identification:** Apply Tarjan's conditions.
5. **Experimental Verification:** Remove identified critical points and measure the actual connectivity impact.

The implementation achieves optimal $O(|V| + |E|)$ time and space complexity, making it suitable for analyzing massive real-world networks.

7 Description of Experiments and Datasets Used

The project's experimental design involved a preliminary testing phase, followed by a large-scale analysis of six real-world network datasets.

Table 7: Six real-world network datasets used in the analysis.

Category	Dataset File
Road Networks	`roadNet-CA.txt`
	`roadNet-PA.txt`
Internet Networks	`as20000102.txt`
	`as-skitter.txt`
Power Grids	`power_grid_uci.txt`
	`power-US-Grid.txt`

7.1 Description of Experiments

The experimental framework was constructed to validate and test the implemented algorithm and to perform an exhaustive analysis of network structures.

7.1.1 Experiment 1: Test on Known Benchmark Graphs

The main goal was to use a set of benchmark tests with known structural properties to make sure the algorithm was correct. These tests included the Path Graph (P5), the Star Graph (K1,4), the Complete Graph (K5), the Cycle Graph (C5), the Wheel Graph (W5), and the Petersen Graph. This method proved that the algorithm was correct for a wide range of topologies.

7.1.2 Experiment 2: Analyze Sensitivity to Graph Density

This experiment looked at how network connectivity and structural resilience are related. We started with a minimal spanning tree with 49 edges and added edges one at a time until the graph became biconnected (no bridges).

7.1.3 Experiment 3: Measure Runtime with Increasing Graph Complexity

This experiment validated the algorithm's theoretical $O(n + m)$ time complexity on the six real-world datasets. The execution time for detecting both the AP and bridges was measured and reported in one go.

8 Results and Discussion

The application of the validated algorithm to all datasets yielded quantifiable results across several metrics.

8.1 Correctness and Validation

We tested the algorithm's accuracy on benchmark graphs. The algorithm's accuracy across all types of network vulnerabilities and resilience patterns was confirmed by passing all benchmark tests.

Table 8: Benchmark graph validation results

Graph Type	**Nodes (n)**	**Articulation Points**	**Bridges**
Path Graph (P_5)	5	3	4
Star Graph ($K_{1,4}$)	5	1	4
Complete Graph (K_5)	5	0	0
Cycle Graph (C_5)	5	0	0
Wheel Graph (W_5)	5	0	0
Petersen Graph	10	0	0

8.2 Analysis of Sensitivity to Graph Density

An experiment on a 50-node random graph began with a sparse structure of 49 edges (density 0.040), which contained 23 bridges. As edges were added, the number of bridges declined. The graph became fully biconnected (0 bridges) once it contained 110 edges, corresponding to a density of 0.090. This demonstrates that a modest increase in connectivity, from a density of 4.0% to 9.0%, can eliminate all single-edge vulnerabilities.

Table 9: Reduction in Bridges with Increasing Graph Density

Number of Edges	**Graph Density**	**Bridges Found**
49	0.040	23
59	0.048	21
70	0.057	9
82	0.067	4
92	0.075	2
105	0.086	1
110	0.090	0

8.3 Runtime with Increasing Graph Complexity

The algorithm's performance to measure time taken to detect both APs and bridges simultaneously scaled linearly with graph size $(n + m)$, as expected. For the `power-US-Grid.txt` network, the analysis completed in 0.0053 seconds. For the largest graph, `as-skitter.txt`, the runtime was 6.6877 seconds, validating the $O(n + m)$ complexity.

Table 10: Performance Results Showing O(V+E) Time Complexity Scaling

Dataset	**n+m**	**Detection Time(s)**	**Time/(n+m) μs**
`power-US-Grid.txt`	11,536	0.0053	0.460
`power_grid.uci.txt`	14,968	0.0071	0.474
`as20000102.txt`	20,369	0.0064	0.314
`roadNet-PA.txt`	4,171,888	1.9748	0.473
`roadNet-CA.txt`	7,498,420	3.8465	0.513
`as-skitter.txt`	12,791,713	6.6877s	0.523

Considering the data in Table 10, we note that the algorithm maintains **constant average time per element** (0.3-0.5 μs) across three orders of magnitude in network size. This **linear scaling relationship** between processing time and graph size (n+m) empirically validates the theoretical O(V+E) time complexity.

8.4 Network Robustness Analysis: Critical Point Verification

The analysis of six real-world network datasets reveals significant differences in structural robustness, with internet networks demonstrating superior resilience compared to road networks and

power grids. This robustness can be quantitatively demonstrated by examining the **fraction of articulation points (APs)** relative to total vertices across different network types.

Quantitative Analysis of Network Fragility

The fragility of each network type becomes evident when calculating the percentage of vertices that serve as critical articulation points:

- **Road Networks:** Extremely fragile with high AP concentrations
  - `roadNet-CA.txt`: $\frac{327,864}{1,965,206} = 16.68\%$ of vertices are APs
  - `roadNet-PA.txt`: $\frac{193,774}{1,088,092} = 17.81\%$ of vertices are APs
- **Internet Networks:** Remarkably robust with low AP fractions
  - `as-skitter.txt`: $\frac{111,541}{1,696,415} = 6.58\%$ of vertices are APs
  - `as20000102.txt`: $\frac{600}{6,474} = 9.27\%$ of vertices are APs
- **Electrical Power Grids:** Highly vulnerable with substantial AP presence
  - `power_grid.uci.txt`: $\frac{1,655}{6,659} = 24.85\%$ of vertices are APs
  - `power-US-Grid.txt`: $\frac{1,229}{4,941} = 24.87\%$ of vertices are APs

Internet Network Superiority

The internet networks demonstrate exceptional robustness, with `as-skitter.txt` having only **6.58%** of its vertices as articulation points—less than **half** the fragility of road networks and approximately **one-fourth** the vulnerability of power grids.

The contrast becomes even more apparent when comparing bridge densities and the network's ability to maintain connectivity despite multiple critical point failures. Internet networks are designed with redundancy and multiple pathways as fundamental principles, resulting in their superior performance under stress conditions.

Table 11: Summary of Critical Points Identified Across All Datasets

Dataset	Vertices	Edges	APs	Bridges
`roadNet-CA.txt`	1,965,206	5,533,214	327,864	376,517
`roadNet-PA.txt`	1,088,092	3,083,796	193,774	216,994
`as20000102.txt`	6,474	13,895	600	2,451
`as-skitter.txt`	1,696,415	11,095,298	111,541	232,141
`power_grid.uci.txt`	6,659	8,309	1,655	2,736
`power-US-Grid.txt`	4,941	6,595	1,229	1,611

Fractional Analysis Summary

Network Type	AP Fraction	AP Percentage	Robustness Ranking
`as-skitter.txt` (Internet)	0.0658	6.58%	1 (Most Robust)
`as20000102.txt` (Internet)	0.0927	9.27%	2
`roadNet-CA.txt` (Road)	0.1668	16.68%	3
`roadNet-PA.txt` (Road)	0.1781	17.81%	4
`power_grid.uci.txt` (Power)	0.2485	24.85%	5
`power-US-Grid.txt` (Power)	0.2487	24.87%	6 (Least Robust)

The fractional analysis clearly demonstrates that internet networks maintain their structural integrity with significantly fewer critical points per vertex, explaining their superior robustness in real-world scenarios.

The analysis of the six datasets revealed three distinct classes of network structures, with criticality verified through targeted removal experiments.

- **Road Networks:** Found to be extremely fragile (`roadNet-CA.txt` yielded 327,864 APs and 376,517 bridges). Explicit verification showed bridge removal created 2 components, while AP removal could create thousands (e.g., 2,639 in the CA network).
- **Internet Networks:** Proved far more resilient. The `as-skitter.txt` graph (111,541 APs) fragmented into 757 components after an AP removal, indicating robust pathways but also critical hubs.
- **Electrical Power Grids:** Vulnerable with tree-like structures, with the details being discussed in section 3.1 of this report.

8.5 Specific Critical Point Removal Impacts

The experimental verification revealed evidence of network fragmentation of varying magnitude.

Table 12: Detailed impacts of specific critical point removals

Network	Critical Point Type	Components Before	Components After
`roadNet-PA.txt`	Bridge (111427, 111429)	1	2
`roadNet-PA.txt`	Articulation Point 689085	1	207
`roadNet-CA.txt`	Bridge (382844, 382845)	1	2
`roadNet-CA.txt`	Articulation Point 1061877	1	2639
`as20000102.txt`	Bridge (76, 6422)	1	2
`as20000102.txt`	Articulation Point 1119	1	2
`as-skitter.txt`	Bridge (759712, 1464019)	1	2
`as-skitter.txt`	Articulation Point 1095748	1	757
`power_grid_uci.txt`	Bridge (116, 4141)	1	2
`power_grid_uci.txt`	Articulation Point 1260	1	2
`power-US-Grid.txt`	Bridge (4193, 4192)	1	2
`power-US-Grid.txt`	Articulation Point 3895	1	2

9 Applications of Critical-Points-Detecting-Algorithms to Network Reliability

Network dependability is largely about how well a system can remain working when some of its elements break. Our structural approach empirically illustrates that dependability is not merely contingent upon component quantity, but rather emerges from specific network/topological configurations and their inherent vulnerability distributions. Identifying and categorizing important sites facilitates the transition from reactive maintenance to predictive resilience engineering.

9.1 Quantitative Vulnerability Assessment and Failure Characterization

Our empirical findings in the sub-project (**Section 3.1**) indicate two fundamentally unique failure mechanisms with substantial implications for reliability engineering:

1. **Bridges for Binary Disconnection:** It is theoretically foreseeable that when a bridge fails, it will always break into two parts (which happened in 100% of the 2,736 power grid bridge failures). This binary fragmentation pattern lets network operators plan for the worst-case situation of a single link failure, which helps them create good plans for what to do next.
2. **Variable Fragmentation (Articulation Points):** Most articulation points don't do much harm, but a few do a lot of damage (**Section 3.1**). This is because they have a power-law impact distribution. The power grid research found that only one node (0.06% of APs) had a big effect, whereas 61% had a small effect. This distribution shows that dependability efforts should focus on finding and protecting the important few instead of the unimportant many.

Adding Largest Connected Component (LCC) retention as a major indicator helps us understand how strong a network is. The LCC retention rate for the electrical grid was an incredible 99.62% in the worst situation. This means that even as things become broken apart, the main parts still work the same way. This distinction between losing connectivity and losing functionality is crucial for prioritizing reliability investments.

Indeed, a good approach to save money is to take down bridges, which stops network splitting. If the power system (**Section 3.1**) protected or duplicated just 2,736 important links (32.9% of edges), it would get rid of all single-point connection failures. On the other side, fixing the 1,655 APs would cost a lot more with a rather poor return on investment.

9.2 Preventing Cascading Failures and Making Systems More Resilient at Several Points

The multi-point failure analysis – in **Section 3.1**, i.e. the subproject – demonstrates that the largest concern with reliability is failure synergy. The electricity grid simulation we performed revealed that:

- When three APs failed at the same time, they created up to 20 pieces. This is a 58.3% increase over the worst single-point failure.
- The average fragmentation of 6.25 components is 380% (**independent study – whose source code is not attached to this project – conducted by us**) greater than what is expected for strikes that only hit one point. This non-linear damage escalation demonstrates that conventional single-failure analysis significantly underestimates actual vulnerability.

This empirical evidence implies a transformation in reliability engineering from component-level to system-level analysis. Now, protection plans need to think about:

- **Geographic Correlation:** Guarding against hurricanes, earthquakes, and other regional phenomena that affect more than one important site at once.
- **Functional Interdependence:** Showing how faults in one aspect of a network (like communication) can affect other portions (like power distribution).
- **Adaptive Topology:** Adding dynamic reconfiguration tools that can keep connections going even when several things go wrong at once.

9.3 Wider Effects on Protecting Important Infrastructure

The methodology and findings of this study create a quantitative foundation for reliability engineering across diverse infrastructure sectors:

- **Communication Networks:** The internet's robust topology (meaning that failures at one place don't generate a lot of fragmentation) shows that its design ideas could be valuable in other parts of infrastructure. Finding important hubs, such as nodes that produce 757 pieces in AS-Skitter, illustrates where there are weak points that need to be secured.
- **Transportation Systems:** Road networks are very fragile since they include thousands of parts that can break at once. This highlights how crucial it is to develop alternative paths, especially for important supply lines and evacuation routes.
- **Emergency Management:** The multi-point failure analysis gives emergency planners numbers that show how much harm might happen in different failure situations. This helps them figure out where to put resources and how to respond better.
- **Regulatory Framework:** These findings advocate for the formulation of evidence-based dependability standards that require specific topological resilience measures instead of broad redundancy criteria.

Moving from qualitative vulnerability assessment to quantitative risk prediction is a big step forward in reliability engineering. Network operators can use their limited resources to make their networks more reliable by figuring out which parts are most important and how failures spread. This data-driven approach goes beyond standard safety factors and general redundancy to a predictive,

topology-aware resilience model that looks at how current infrastructure systems are connected and how complex they are.

10 Conclusion

This project's analysis, conducted with a verified algorithm, provides a comprehensive, data-driven summary of the structural integrity of several major real-world networks. The results confirm the algorithm's $O(n+m)$ efficiency and its suitability for large-scale analysis. The explicit verification experiments revealed fundamental differences in the fragility of various infrastructure types.

11 Challenges Faced and Lessons Learned

11.1 Technical Obstacles and Solutions

- **Recursion Depth Management:** The primary obstacle was Python's default recursion limit. Analyzing large networks like `roadNet-CA.txt` caused a `RecursionError`. This was solved by explicitly adjusting the system limit to a high margin (`sys.setrecursionlimit(2000000)`).
- **Proof of Concept Efficiency:** Designing the explicit verification step required rebuilding the graph for each test, which was a critical test of memory management and confirmed that the $O(n+m)$ complexity holds even under intensive reconstruction.

11.2 Lessons Learned

1. **The Supremacy of Asymptotic Bounds:** The validation of the $O(n+m)$ complexity was crucial, confirming Tarjan's algorithm is provably optimal and highly efficient in practice.
2. **Fundamental Difference Between Bridge and Articulation Point Impact:** Experimental results revealed a crucial distinction: bridge removal consistently increases the number of connected components by exactly one, creating a binary split. In contrast, articulation point removal can fragment networks into hundreds or thousands of components.
3. **Redundancy is Measurable and Targeted:** The density analysis demonstrated that adding just enough edges to form cycles (reaching ≈ 0.090 density) eliminates all single points of failure. The predictable behavior of bridges provides a clear optimization target.
4. **Correctness Requires Empirical Verification:** The decision to computationally prove that the identified points were genuinely critical reinforced the rigor necessary for high-stakes infrastructure analysis.

Resources

The datasets used in this analysis were sourced from the following repositories:

- **Road Networks:**
  - California Road Network (`roadNet-CA.txt`): `https://snap.stanford.edu/data/roadNet-CA.html`
  - Pennsylvania Road Network (`roadNet-PA.txt`): `https://snap.stanford.edu/data/roadNet-PA.html`
- **Internet Networks:**
  - Autonomous Systems (`as20000102.txt`): `https://snap.stanford.edu/data/as-733.html`
  - Internet Topology (`as-skitter.txt`): `https://snap.stanford.edu/data/as-Skitter.html`
- **Power Grids:**

* US Power Grid (`power-US-Grid.txt`): `https://networkrepository.com/power-US-Grid.php`
* Power Grid UCI (`power_grid_uci.txt`): From the UCI Network Data Repository

Citation for Network Repository:

```
@inproceedings{nr,
title={The Network Data Repository with Interactive
Graph Analytics and Visualization},
author={Ryan A. Rossi and Nesreen K. Ahmed},
booktitle={AAAI},
url={https://networkrepository.com},
year={2015}
}
```

KINDLY CLICK HERE TO BE DIRECTED TO THE GITHUB PROJECT REPOSITORY