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 6 (Time and Space Complexity Analysis), Section 8 (Code Implementation Overview), and Section 9 (Results and Discussion).*
- **Task 2:** Validate results on test cases with known bridges and articulation points.
  - *This is delineated in Section 8 (Description of Experiments) and Section 9 (Results and Discussion).*
- **Task 3:** Discuss applications in network reliability and clustering.
  - *This is delineated in Section 10 (Applications of Critical-Points-Detecting-Algorithms to Network Reliability).*

Primary Objective: Experiment Completion The source code of the primary objective is in the GITHUB repository and is titled `Primary_Project_SomdasTeam.py`. The corresponding output for this script is in the PDF file titled `Primary_Project_SomdasTeam_Outpt.pdf`.

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 8 (Description of Experiments and Datasets Used) and Section 9 (Correctness and Validation).*
- **Experiment 2:** Analyze sensitivity to graph density.
  - *This is delineated in Section 9 (Analysis of Sensitivity to Graph Density).*
- **Experiment 3:** Measure runtime with increasing graph complexity.
  - *This is delineated in Section 9 (Runtime with Increasing Graph Complexity) and Section 10 (Network Analysis Robustness and Analysis).*

Secondary Objective: Sub-Project Completion The source code of the secondary objective is in the GITHUB repository and is titled `Secondary_Project_SomdasTeam.py`. The corresponding output is in the PDF file titled `Secondary_Project_SomdasTeam_Output.pdf`.

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 4** (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, Preliminaries, 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 of them from different sources online.

For the sub-project, we used variants of the primary source code, titled "Secondary-Project-SomdasTeam.py", which are mere improvisations of the same, and hence are not shared. However, the primary source code is shared on the GITHUB repository. Nevertheless, the outputs of those variants are shared on the GITHUB repository. They are titled: "Average-Fragmentation-Vs-AP-Multipoint Attack.pdf" and "Trial-Runs-10-AP-Multipoint Attack.pdf".

The filenames of the SIX REAL-WORLD DATASETS of graphs are the following: `as20000102.txt`, `as-skitter.txt`, `power_grid_uci.txt`, `power-US-Grid.txt`, `roadNet-CA.txt`, and `roadNet-PA.txt`. The source of all these datasets is explicitly given in Section 12, Resources and in the GITHUB repository, whose link is shared in the last page of this report.

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 units 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.

For instance, geography and high capital investment often limit physical infrastructures networks and graphs, 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 Impact of 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 4).

– **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` (a real-world network dataset that this project will analyse in a later section), 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` (a real-world network dataset that this project will analyse in a later section), 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 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.

4 Secondary Objective of this Project: Finding Critical Points in a Power Grid Whose Failure Would Split the System

The source code of the secondary objective is in the GITHUB repository and is titled `Secondary_Project_SomdasTeam.py`. The 0output is in the PDF file titled: `Secondary_Project_SomdasTeam_Output.pdf`.

Please note: For the sub-project, we used variants of the primary source code, titled "Secondary-Project-SomdasTeam.py", which are mere improvisations of the same, and hence are not shared. However, the primary source code is shared on the GITHUB repository. Nevertheless, the outputs of those variants are shared on the GITHUB repository. They are titled: "Average-Fragmentation-Vs-AP-Multipoint Attack.pdf" and "Trial-Runs-10-AP-Multipoint Attack.pdf".

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 analysis was done with a Python script that used the networkz library.

4.1 Multi-Point Failure Vulnerability Analysis

Experimental Objectives and Methodology: This section presents a systematic analysis of coordinated multi-point failures in power grid infrastructure through large-scale simulation experiments. The primary objective is to quantitatively characterize the relationship between the number of articulation points removed and resulting network fragmentation, establishing empirical vulnerability scaling laws for critical infrastructure protection.

The experimental design implements progressive escalation from single-point to coordinated 50-point failure scenarios, conducting 100 independent random trials at each removal level (1, 3, 5, 7, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50 APs) to ensure statistical reliability. Additional validation experiments conducted 100 to 40,000 trials for 10-AP removal scenarios to assess metric stability across sampling scales.

So, our primary objective of this subproject include:

- Quantifying fragmentation scaling relationships from individual to coordinated attacks
- Establishing statistical confidence intervals for vulnerability assessments
- finding the thresholds for protecting critical infrastructure
- Checking to see if simulation-based risk analysis can accurately predict outcomes

The following analysis presents empirical evidence from 2,500+ individual failure simulations across 25 coordinated attack scenarios, providing comprehensive vulnerability characterization for power grid resilience planning.

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

4.1.1 Vulnerability Assessment of the (`power_grid_uci.txt`)Network

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 very LCC retention (%).

(**LCC Retention (%)** is a metric used in network science to quantify how well a network maintains its core connectivity after a failure, such as the removal of a node or edge. It measures the size of the Largest Connected Component (LCC) that remains after the failure, expressed as a percentage of the network that could have possibly remained connected.)

Table 2: Impact Analysis of Top 5 Most Critical Nodes, LCC Retention (%) defined in the following paragraph

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 Retention %**
(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 retention (%) indicates a network architecture characterized by significant structural redundancy and robust connectivity patterns. 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.

The practical significance of this finding lies in the network's ability to maintain operational integrity under stress conditions. With LCC retention (%) 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 Analysis: Empirical Evidence from Systematic AP Removal

Experimental Protocol To assess the network's resilience under more realistic, complex failure scenarios, a systematic multi-point failure analysis was conducted. This experiment involved the simultaneous random removal of an increasing number of articulation points (APs), from 1 to 50. For each distinct removal level, 100 independent trials were executed, and the resulting network fragmentation was measured by counting the number of connected components. This methodology allows for the quantification of synergistic effects and the identification of a strongly linear scaling in vulnerability.

Indeed, the experimental results demonstrate a clear and predictable escalation in network fragmentation as simultaneous AP failures increase. The relationship between APs removed and average fragmentation follows a strongly linear pattern, modeled by the equation:

$$\textbf{Average Fragments} = \textbf{1.74} \times \textbf{(APs removed)} + \textbf{0.16} \quad (\mathrm{R}^2 = 0.998)$$

Quantitative Multi-Point Failure Analysis The results, presented in Table 4, demonstrate a clear and predictable escalation in network damage as the number of simultaneous failures increases. The relationship between the number of APs removed and the average fragmentation is strongly linear, modeled by the equation **Average Fragments = 1.74 × (APs removed) + 0.16**, with an R-squared value of $R^2 = 0.998$. This indicates that, on average, each additional AP failure contributes approximately 1.74 new fragments.

The synergistic effect is evident when comparing discrete levels of attack. A 3-AP removal yields an average of 6.19 fragments, whereas a 10-AP removal yields 18.72 fragments—an increase in fragmentation of 187%, far greater than the increase in removed nodes. This linear damage escalation highlights that conventional single-failure analysis significantly underestimates the network's vulnerability to coordinated events. The worst-case (maximum) and best-case (minimum) fragmentation scenarios also scale predictably, establishing clear upper and lower bounds for damage assessment at each attack level.

Statistical Stability Evidence To validate the reliability of the simulation, the stability of the statistical metrics was tested by varying the number of trials for a fixed 10-AP removal scenario.

As shown in Table 5, the average number of fragments remained remarkably stable, varying by only 1.2% (from 17.86 to 18.08) across scales from 100 to 40,000 trials. This demonstrates that a sample of 1,000 trials is more than sufficient for achieving a reliable estimate of the average outcome. Furthermore, the extreme values (Max and Min fragments) converge, indicating that the simulation effectively captures the probable bounds of network damage.

Empirical Risk Assessment and Engineering Implications This quantitative analysis provides a direct framework for risk assessment and engineering decisions. While the network is resilient to single-point failures (1 AP removal creates only 2.58 fragments, affecting less than 0.1% of the network),

Table 4: Empirical Fragmentation Scaling with Increasing Coordinated Failures (100 trials per data point)

APs Removed	Avg Fragments	Max Fragments	Min Fragments
1	2.58	8	2
3	6.19	12	4
5	9.86	19	6
7	12.83	29	7
10	18.72	32	12
12	21.70	34	14
14	23.90	38	16
16	28.51	47	19
18	31.56	42	21
20	35.14	49	25
22	38.51	54	25
24	42.40	59	29
26	45.40	66	36
28	49.61	69	34
30	51.14	67	37
32	54.83	81	40
34	58.33	73	45
36	62.69	87	44
38	65.71	85	49
40	69.42	94	54
42	73.21	99	58
44	76.21	95	59
46	79.99	109	59
48	84.06	105	66
50	86.99	107	66

Table 5: Fragmentation Metric Stability for 10-AP Removal Scenario

Trial Count	Avg Fragments	Max Fragments	Min Fragments
100 trials	18.08	35	11
400 trials	17.86	31	11
1,000 trials	17.95	31	11
11,000 trials	18.08	37	10
40,000 trials	18.05	38	10

it is clearly vulnerable to coordinated attacks. The predictable linear scaling allows for accurate risk projection; for example, one can reliably predict that a simultaneous failure of 15 APs would result in approximately 26-27 fragments.

The key engineering implications are twofold:

1. The linear fragmentation scaling factor of **1.74 fragments per AP** provides a quantitative basis for infrastructure protection planning, allowing for a cost-benefit analysis of hardening a given number of critical nodes.
2. The predictable escalation from 3-AP to 10-AP attacks (a 187% increase in fragmentation) empirically demonstrates measurable synergistic effects, proving that reliability models must account for multi-point failure scenarios to be accurate.

The bounded and stable fragmentation ranges establish known vulnerability limits, transitioning risk assessment from a qualitative exercise to a predictive, data-driven science.

4.2 Conclusion and Recommendations

4.2.1 Systemic Vulnerability Assessment and Future Research Directions

Empirical Vulnerability Characterization: The quantitative analysis of 2,500+ multi-point failure simulations reveals that the principal vulnerability of the `power_grid_uci` network is not catastrophic single points of failure, but rather its **systemic fragility under coordinated multi-point attacks**. The experimental evidence demonstrates:

- **Distributed Critical Infrastructure:** 1,655 articulation points distributed throughout the network create pervasive vulnerability points
- **Linear Scaling Threat:** Coordinated attacks follow predictable fragmentation scaling (1.74 fragments per AP removed)
- **Measurable Synergistic Effects:** 10-AP removal produces 187% more fragmentation than 3-AP removal (18.72 vs 6.19 average fragments)
- **Statistical Reliability:** Fragmentation metrics stabilize with 1,000+ trials.

Quantitative Risk Profile:

- **Single-Point Resilience:** Maximum 8 fragments from worst single AP removal (0.16% of network)
- **Multi-Point Vulnerability:** Maximum 109 fragments from 46 AP removal (2.21% of network)
- **Predictable Escalation:** Linear relationship enables accurate risk projection for coordinated attacks
- **Bounded Impact:** 10-AP attacks consistently produce 10-38 fragments, establishing known vulnerability limits

Research Limitations and Validation: The current analysis provides comprehensive structural vulnerability assessment through 25 coordinated attack scenarios and 2,500+ simulations. However, several dimensions remain unexplored due to computational and data constraints:

- **Flow and Capacity Integration:** Current graph-theoretic analysis assumes uniform node importance. Integration of power flow data and node capacity would reveal functional criticality beyond structural analysis. A "Low-Impact" AP serving high-capacity transmission would demonstrate significantly higher operational criticality.
- **Targeted Multi-Point Simulation:** Random AP removal provides statistical baseline; targeted removal of top-N critical APs (nodes 3088, 1709, 2117) would model worst-case coordinated attacks. Geographically proximate AP removal would simulate localized disaster scenarios with potentially amplified regional impact.
- **Component Size Distribution Analysis:** Beyond largest connected component tracking, full fragment size distribution analysis would characterize whether failures produce balanced fragmentation or isolated single-node "dust" components, informing recovery prioritization strategies.
- **Temporal Cascading Dynamics:** Current static analysis assumes simultaneous failures; sequential failure simulation would model realistic cascading scenarios with time-dependent recovery interventions.

Engineering Significance: The established linear fragmentation scaling (1.74 fragments/AP) and statistical stability across trial counts provide reliable quantitative foundation for infrastructure protection planning. The predictable vulnerability escalation from individual to coordinated attacks enables cost-effective defense prioritization against measurable multi-point threat scenarios.

By integrating these additional analytical dimensions with the established structural vulnerability baseline, future research can develop comprehensive, operationally relevant resilience models for critical infrastructure protection.

5 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 source code of the primary objective is in the GITHUB repository and is titled `Primary_Project_SomdasTeam.py`. The corresponding output for this script is in the PDF file titled `Primary_Project_SomdasTeam_Output.pdf`.

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.

6 Time and Space Complexity Analysis of Tarjan's Algorithm

6.1 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).

6.2 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.
- **node_to_index, index_to_node:** Hash maps for node-ID mapping, size $O(n)$

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.

6.3 Time Complexity Analysis

6.3.1 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.

6.3.2 Node-ID Mapping Construction

The algorithm creates `node_to_index` and `index_to_node` dictionaries by iterating through all nodes. This requires $O(n)$ time.

6.3.3 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 exactly 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 $O(m)$.
- **Node-ID conversion:** Each edge traversal requires $O(1)$ hash map lookups for node-ID conversion, contributing $O(m)$ total.

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

6.3.4 Post-Processing

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

6.3.5 Component Counting Functions

The functions `count_components_after_removal()` and `count_components_after_bridge_removal()` perform BFS/DFS over all vertices and edges, requiring $O(n + m)$ time each. However, these are called separately for experimental analysis and do not affect the core algorithm complexity.

6.3.6 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.

6.4 Space Complexity Analysis

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

Table 6: 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	$O(n)$
node_to_index	Hash map for node-ID conversion	$O(n)$
index_to_node	Hash map for reverse node-ID mapping	$O(n)$
Recursion stack	Maximum depth can be up to n	$O(n)$

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

6.5 Algorithmic Correctness and Implementation Notes

The implementation correctly handles:

- **Disconnected graphs:** Multiple DFS traversals from unvisited nodes
- **Node-ID mapping:** Handles arbitrary node labels through hash maps
- **Edge cases:** Empty graphs, single-node graphs, and trees
- **Recursion depth:** Explicit recursion limit adjustment for large graphs

6.6 Empirical Validation

The complexity analysis is empirically validated by the experimental results:

- **Linear scaling:** Processing time grows linearly with $n + m$ across all test networks
- **Large-scale efficiency:** Successfully handles networks with 1.6+ million nodes
- **Memory efficiency:** Consistent performance without exponential memory growth

The algorithm demonstrates optimal $O(n+m)$ complexity in both theory and practice, making it suitable for large-scale network analysis.

6.7 Additional Considerations

6.7.1 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.

6.7.2 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 7: 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	$O(n+m)$	$O(n+m)$

6.8 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.

7 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.

7.1 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.

7.2 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.

7.3 Algorithm Workflow

The implementation follows this precise workflow:

- **Graph Construction:** Build an adjacency list from the input data.
- **DFS Initialization:** Initialize discovery time, low-link values, and parent arrays.
- **Recursive DFS Traversal:** Process each vertex, updating low-link values and checking for bridge/AP conditions.
- **Critical Point Identification:** Apply Tarjan's conditions.
- **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.

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)** μ**s**
`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:** These were found to be fragile. The 'roadNet-CA.txt' dataset yielded 327,864 articulation points and 376,517 bridges. The high ratio of critical points to total nodes and edges confirms a topology dominated by single-point failures.
- **Internet Networks:** Proved far more resilient. The `as-skitter.txt` graph (111,541 APs) fragmented into 757 components after a particular AP removal, indicating robust pathways but also critical hubs. These proved far more resilient. The 'as-skitter.txt' graph, despite having over 1.6 million nodes, contained 111,541 articulation points. While large, this number is proportionally much lower than in road networks, which points to a robust core with many redundant paths.
- **Electrical Power Grids:**These networks were the most vulnerable – according to our investigation. The 'power-US-Grid.txt' dataset, with only 4,941 nodes, had 1,229 articulation points and 1,611 bridges. The high number of bridges relative to its small size demonstrates a heavy dependence on single transmission lines, making it structurally prone to outages.

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 4**) indicate two fundamentally unique failure mechanisms with substantial implications for reliability engineering:

- **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.
- **Variable Fragmentation (Articulation Points):** Most articulation points don't do much harm, but a few do a lot of damage (**Section 4**). 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 4**) 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. Simply put, the capital investment in building a power-station/power-plant, an AP, is far more than connecting two nodes through redundancies, i.e. wires and cables.

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

The multi-point failure analysis - in Section 4, i.e. the subproject - demonstrates that the largest concern with reliability is a vulnerability to multi-pronged attacks, i.e. several APs being compromised simultaneously. 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 linear damage escalation demonstrates that conventional single-failure analysis significantly underestimates actual vulnerability.
- In the escalated 10-AP simultaneous failure scenario, the network fragmented into up to 31 components, representing a 138.5% increase over the worst single-point failure. The average fragmentation of 17.95 components further underscores the linear escalation of damage in coordinated multi-point attacks.

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:** According to extensive research performed in the field of network resilience, 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. This magnitude of fragility is reflected by our analyses on the datasets, but we could not delve deeper given the constraints of our simulations and other factors.

- **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

– **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.

– **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.

– **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.

– **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.

12 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}
}
```

Bibliography

References

[1] Deo, Narsingh (2016). *Graph Theory with Applications to Engineering and Computer Science.* Courier Dover Publications.

[2] West, Douglas B. (2017). *Introduction to Graph Theory* (2nd ed.). Pearson.

[3] Harary, Frank (1969). *Graph Theory.* Addison-Wesley.

[4] van Steen, Maarten (2010). *Graph Theory and Complex Networks: An Introduction.* Maarten van Steen.

[5] Hanish, Natalya P. and Murray, Edward O. and Anderson, David F. (2019). *Network Reliability: Measures and Evaluation.* Wiley.

[6] Clement, Christiane and Owen, Christopher and Vogl, Wolfgang (2021). *Cyber Resilience of Systems and Networks.* Springer.

[7] Cormen, Thomas H. and Leiserson, Charles E. and Rivest, Ronald L. and Stein, Clifford (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.

[8] Kleinberg, Jon and Tardos, Éva (2006). *Algorithm Design.* Pearson.

[9] Bondy, J. A. and Murty, U. S. R. (1976). *Graph Theory with Applications.* Elsevier.

[10] Barabási, Albert-László (2016). *Network Science.* Cambridge University Press.

[11] Massip-da, Antonio G. and Theedoor, G. (2020). *Power Grid Resilience: Assessment and Enhancement.* Wiley.

[12] Ekanayake, Janaka and Jenkins, Nick and Liyanage, Kithsiri and Wu, Jianzhong and Yokoyama, Akihiko (2012). *Smart Grid: Technology and Applications.* Wiley.

[13] Al-Enezi, H.A.A. (2021). *Resilience of De-Centralized Smart-Grid.* Springer.

13 GITHUB LINK

KINDLY CLICK HERE TO BE DIRECTED TO THE GITHUB PROJECT REPOSITORY