– Project Report

Submitted in partial fulfilment of the requirements for the award of degree of

Topic: GraphX Academic Citation Network Analysis

B TECH (CSE)

Submitted To: Dr. Aryan Maan

Submitted By:

Name :-	**Rahul Mishra**
Reg No:-	**12220689**
Roll No :-	**25**
Section :-	**K22GB**
Subject :-	**Cluster Computing**
Course Code :-	**INT315**

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB

Acknowledgement

I would like to express my sincere gratitude to **Dr. Aryan Maan** for valuable guidance and support throughout this project. I am also thankful to Lovely Professional University for providing the facilities and resources to complete this work successfully.

Rahul Mishra

15/11/25

Abstract

Academic research is interconnected through citations. By treating these citations as a graph, meaningful insights can be extracted — such as identifying influential papers, discovering isolated works, and detecting collaborative clusters. In this project, Apache Spark GraphX is used to build and analyze an academic citation network. The analysis includes in-degree calculation, PageRank scoring, isolated-node detection, and network visualization using Python and NetworkX. A dashboard is also created to summarize the results.

Table of Contents

1. Introduction

Citations define how academic research builds upon previous knowledge. Each citation can be represented as a directional link between two papers. When aggregated, these connections form a *citation network*. Network analysis allows:

- Discovery of influential works
- Identification of emerging research areas
- Understanding of collaboration patterns
- Finding isolated or independent research efforts

Apache Spark's **GraphX** is ideal for processing such large-scale graph datasets due to its distributed nature and built-in graph algorithms like PageRank.

This project uses a real-world citation dataset from AMiner (20,000 edges) and performs an end-to-end analysis and visualization pipeline.

2. Problem Statement

Traditional citation analysis tools struggle with:

- Large datasets
- Complex network relationships
- Distributed processing needs

Goal: Build a distributed GraphX pipeline that efficiently analyzes an academic citation network.

3 . Objectives

1. Load and preprocess citation dataset.
2. Construct directed graph using GraphX.
3. Compute in-degree (citation counts).
4. Run PageRank to find influential papers.
5. Identify isolated papers.
6. Visualize the graph using Python.
7. Build a dashboard illustrating important metrics.

4. Technologies Used

Technology	Purpose
Apache Spark	Distributed processing
GraphX	Graph computation engine
Scala	Core backend code
sbt	Build tool
Python	Visualization
NetworkX	Graph drawing
Streamlit	Dashboard creation

5. Methodology

5.1 Dataset Description

The dataset has been derived from AMiner Citation Network V2 and contains:

- 20,000 directed citation edges
- 6,000 papers

5.2 Workflow Steps

- Dataset loaded as RDD
- Graph built using GraphX
- Degrees computed
- PageRank executed
- Isolated nodes detected safely
- Graph exported for visualization
- Python scrip renders the network
- Dashboard summarizes statistics

5.3 GraphX Concepts Used

Vertices

Each paper is a vertex.

Edges

A directed edge exists from A → B if A cites B.

In-degree

Number of incoming edges → number of citations received.

Out-degree

Number of outgoing edges → number of citations made.

PageRank

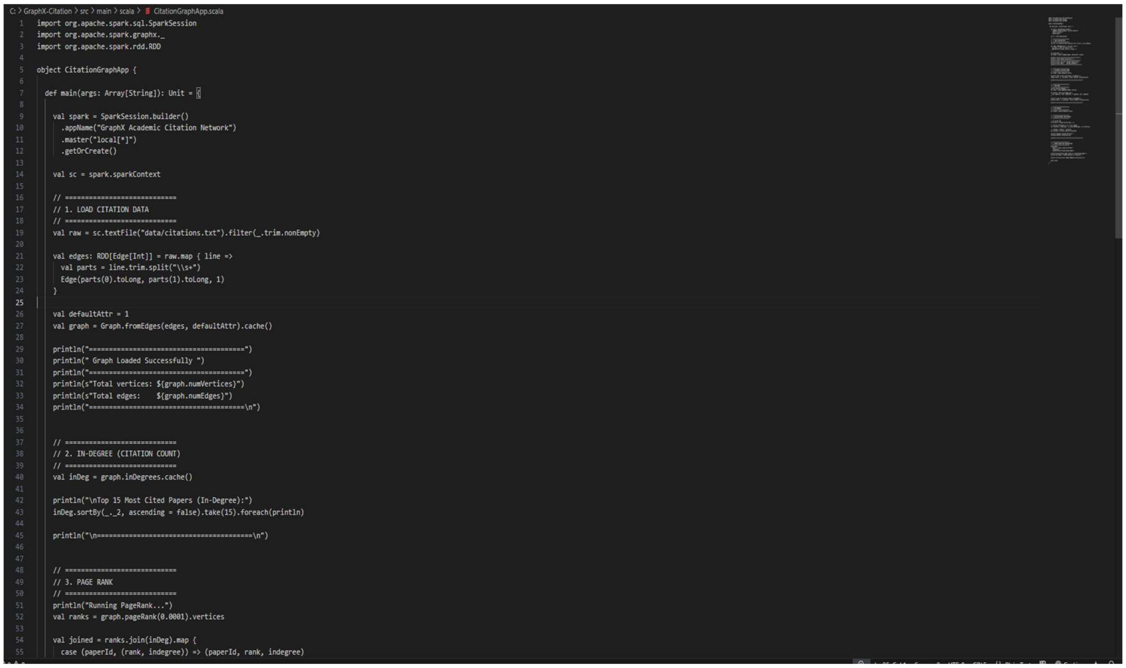
Measures influence using recursive importance weights.

6. Implementation

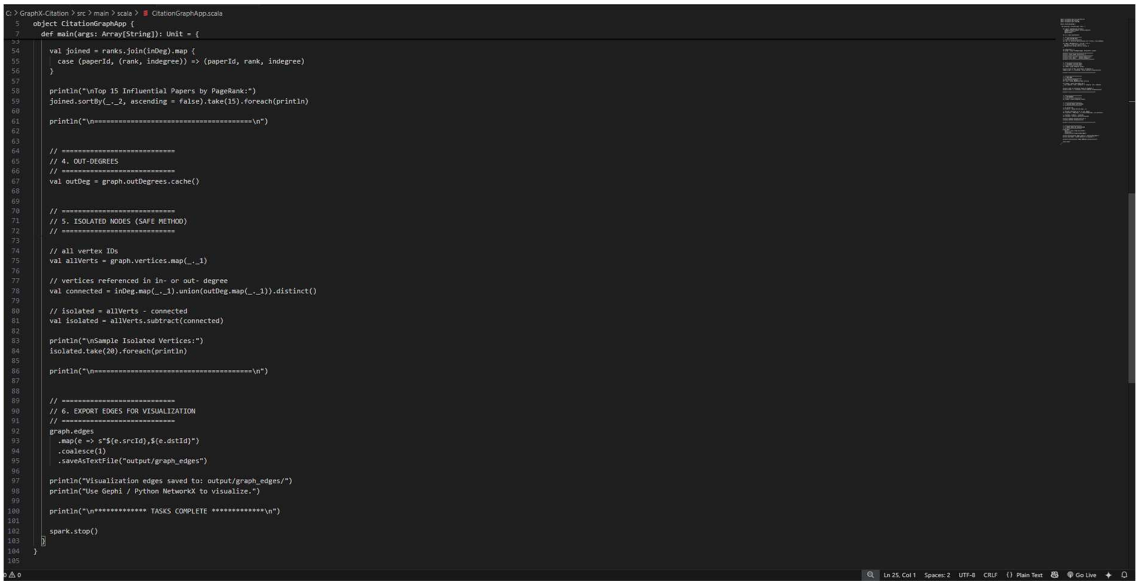
6.1 Project Structure

```
C:\GraphX-Citation\
 ├── build.sbt
 ├── data\
 |    └── citations.txt
 ├── output\
 |    └── graph_edges\
 ├── visualization\
 |     ├── visualize_graph.py
 |    └── dashboard.py
 └── src\
      └── main\
           └── scala\
                └── CitationGraphApp.scala
```

6.2 Scala Code

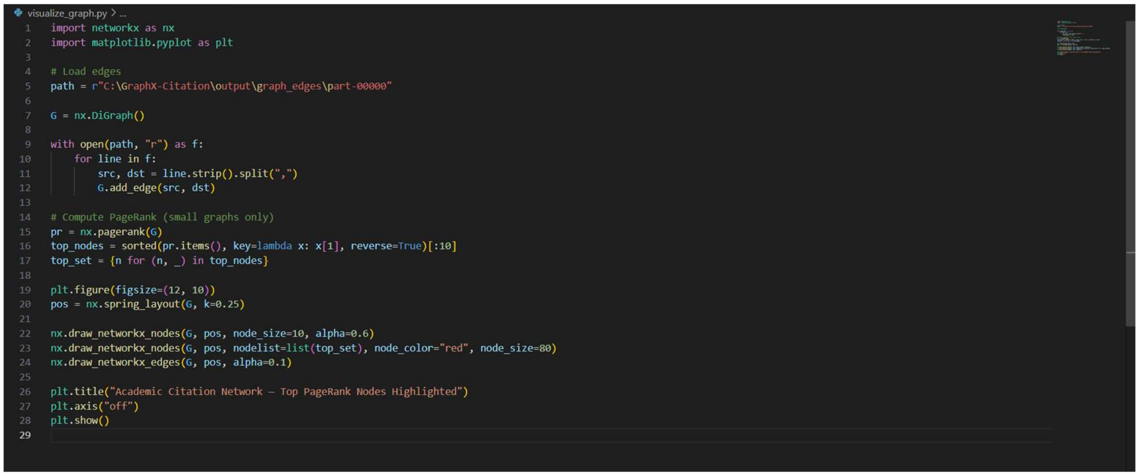

C: > GraphX-Citation > src > main > scala > CitationGraphApp.scala

```scala
import org.apache.spark.sql.SparkSession
import org.apache.spark.graphx._
import org.apache.spark.rdd.RDD

object CitationGraphApp {

  def main(args: Array[String]): Unit = {

    val spark = SparkSession.builder()
      .appName("GraphX Academic Citation Network")
      .master("local[*]")
      .getOrCreate()

    val sc = spark.sparkContext

    // ============================
    // 1. LOAD CITATION DATA
    // ============================
    val raw = sc.textFile("data/citations.txt").filter(_.trim.nonEmpty)

    val edges: RDD[Edge[Int]] = raw.map { line =>
      val parts = line.trim.split("\\s+")
      Edge(parts(0).toLong, parts(1).toLong, 1)
    }

    val defaultAttr = 1
    val graph = Graph.fromEdges(edges, defaultAttr).cache()

    println("========================================")
    println(" Graph Loaded Successfully ")
    println("========================================")
    println(s"Total vertices: ${graph.numVertices}")
    println(s"Total edges:    ${graph.numEdges}")
    println("========================================\n")


    // ============================
    // 2. IN-DEGREE (CITATION COUNT)
    // ============================
    val inDeg = graph.inDegrees.cache()

    println("\nTop 15 Most Cited Papers (In-Degree):")
    inDeg.sortBy(_._2, ascending = false).take(15).foreach(println)

    println("\n========================================\n")


    // ============================
    // 3. PAGE RANK
    // ============================
    println("Running PageRank...")
    val ranks = graph.pageRank(0.0001).vertices

    val joined = ranks.join(inDeg).map {
      case (paperId, (rank, indegree)) => (paperId, rank, indegree)
```


C: > GraphX-Citation > src > main > scala > CitationGraphApp.scala
Ln 25, Col 1
Spaces: 2
UTF-8
CRLF
{} Plain Text
Go Live

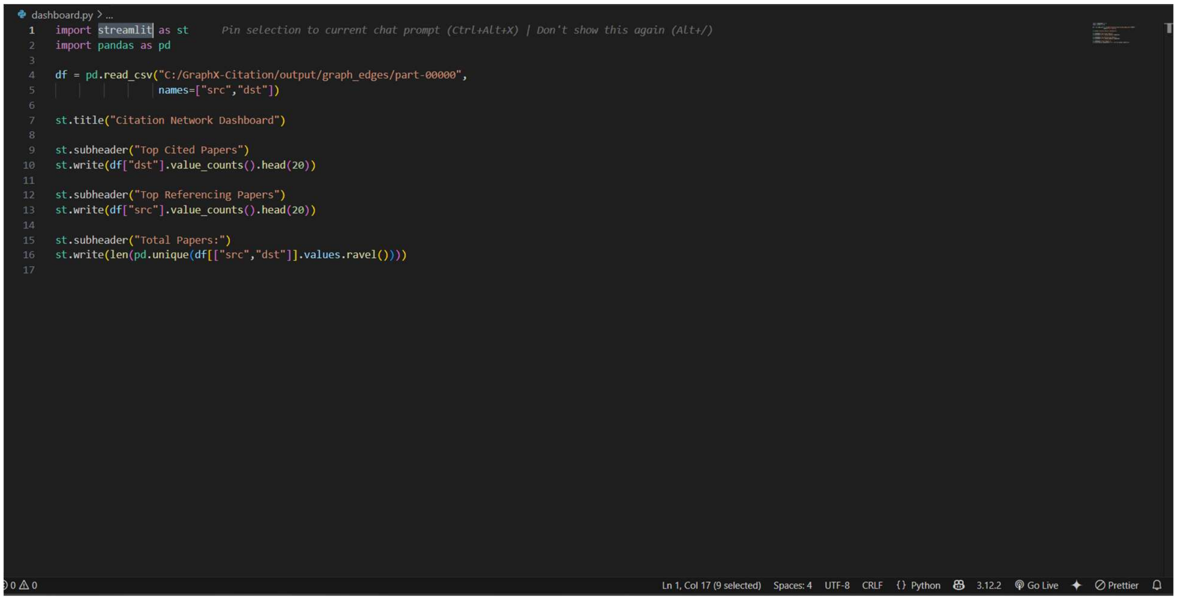
```scala
object CitationGraphApp {
  def main(args: Array[String]): Unit = {
    val joined = ranks.join(inDeg).map {
      case (paperId, (rank, indegree)) => (paperId, rank, indegree)
    }

    println("\nTop 15 Influential Papers by PageRank:")
    joined.sortBy(_._2, ascending = false).take(15).foreach(println)

    println("\n========================================\n")


    // ============================
    // 4. OUT-DEGREES
    // ============================
    val outDeg = graph.outDegrees.cache()


    // ============================
    // 5. ISOLATED NODES (SAFE METHOD)
    // ============================

    // all vertex IDs
    val allVerts = graph.vertices.map(_._1)

    // vertices referenced in in- or out- degree
    val connected = inDeg.map(_._1).union(outDeg.map(_._1)).distinct()

    // isolated = allVerts - connected
    val isolated = allVerts.subtract(connected)

    println("\nSample Isolated Vertices:")
    isolated.take(20).foreach(println)

    println("\n========================================\n")


    // ============================
    // 6. EXPORT EDGES FOR VISUALIZATION
    // ============================
    graph.edges
      .map(e => s"${e.srcId},${e.dstId}")
      .coalesce(1)
      .saveAsTextFile("output/graph_edges")

    println("Visualization edges saved to: output/graph_edges/")
    println("Use Gephi / Python NetworkX to visualize.")

    println("\n************* TASKS COMPLETE *************\n")

    spark.stop()
  }
}
```

6.3 Python Visualization Script



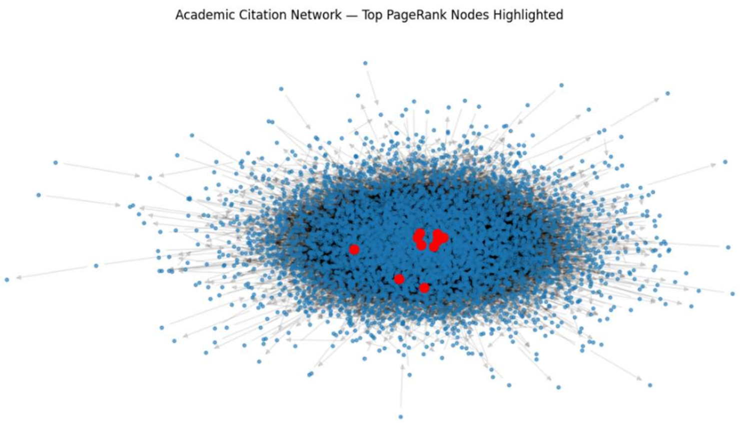
```python
import networkx as nx
import matplotlib.pyplot as plt

# Load edges
path = r"C:\GraphX-Citation\output\graph_edges\part-00000"

G = nx.DiGraph()

with open(path, "r") as f:
    for line in f:
        src, dst = line.strip().split(",")
        G.add_edge(src, dst)

# Compute PageRank (small graphs only)
pr = nx.pagerank(G)
top_nodes = sorted(pr.items(), key=lambda x: x[1], reverse=True)[:10]
top_set = {n for (n, _) in top_nodes}

plt.figure(figsize=(12, 10))
pos = nx.spring_layout(G, k=0.25)

nx.draw_networkx_nodes(G, pos, node_size=10, alpha=0.6)
nx.draw_networkx_nodes(G, pos, nodelist=list(top_set), node_color="red", node_size=80)
nx.draw_networkx_edges(G, pos, alpha=0.1)

plt.title("Academic Citation Network – Top PageRank Nodes Highlighted")
plt.axis("off")
plt.show()
```

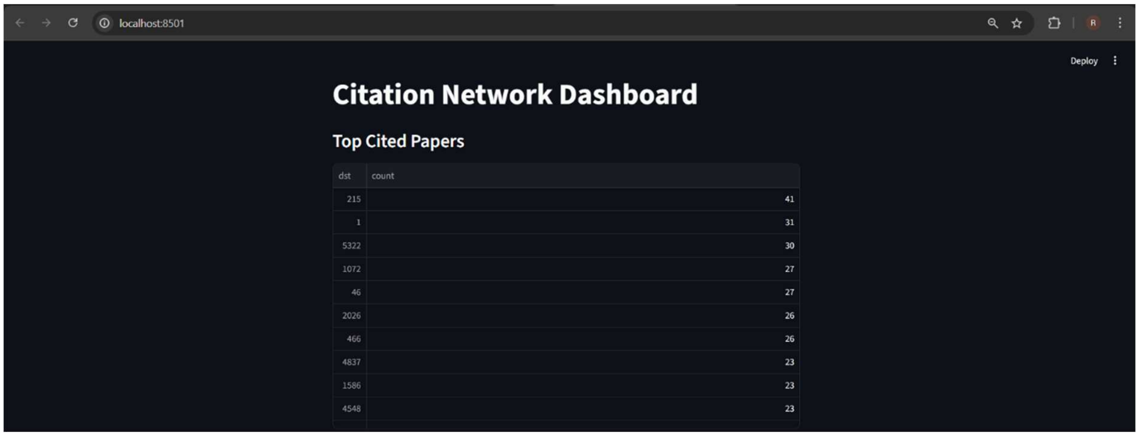
6.4 Dashboard Code


dashboard.py > ...
Pin selection to current chat prompt (Ctrl+Alt+X) | Don't show this again (Alt+/)
0 ⚠ 0
Ln 1, Col 17 (9 selected)
Spaces: 4
UTF-8
CRLF
{ } Python
3.12.2
Go Live
Prettier

```python
import streamlit as st
import pandas as pd

df = pd.read_csv("C:/GraphX-Citation/output/graph_edges/part-00000",
                 names=["src","dst"])

st.title("Citation Network Dashboard")

st.subheader("Top Cited Papers")
st.write(df["dst"].value_counts().head(20))

st.subheader("Top Referencing Papers")
st.write(df["src"].value_counts().head(20))

st.subheader("Total Papers:")
st.write(len(pd.unique(df[["src","dst"]].values.ravel())))
```

7. Results and Screenshots

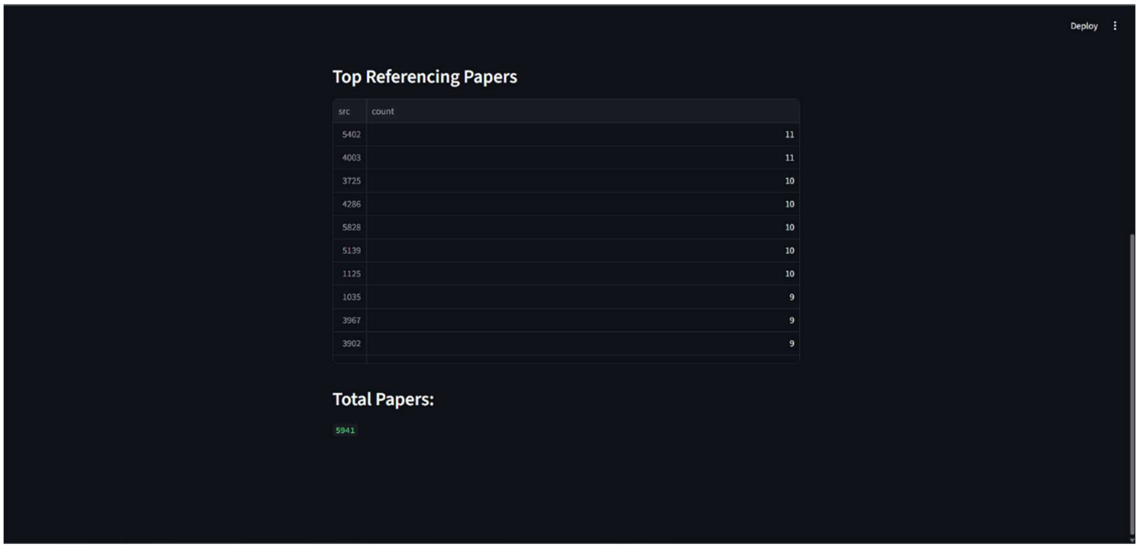




Citation Network Dashboard

Top Cited Papers

dst	count
215	41
1	31
5322	30
1072	27
46	27
2026	26
466	26
4837	23
1586	23
4548	23



Top Referencing Papers

src	count
5402	11
4003	11
3725	10
4286	10
5828	10
5139	10
1125	10
1035	9
3967	9
3902	9

Total Papers:

5941

8. Conclusion

This project successfully demonstrates:

- How Spark GraphX can be used to analyze citation networks
- Performing PageRank on distributed graph data
- Computing academic influence metrics
- Identifying isolated or unique research works
- Visualizing complex networks using Python tools

The end-to-end pipeline is efficient, scalable, and suitable for large datasets.