DOSP Programming Assignment 3
Team 3

Team Information:

Name	UFID	Email
Girish Vinayak Salunke	88908382	gsalunke@ufl.edu
Janhavi Shriram Athalye	76926526	janhavi.athalye@ufl.edu
Reema Solan	98208333	reema.solan@ufl.edu
Srujith Reddy Narra	38866480	narra.sr@ufl.edu

Environment Setup:

1. **Operating System:** This project has been developed and tested on Windows and macOS. It is compatible with any operating system supporting F#.
2. **Required Software:**
   F#, .NET framework
   Visual Studio Code IDE
   AKKA actor framework setup

Compilation:

1. Unzip the PA3_Team3 folder from the zipped packet and open it in Visual Studio Code IDE
2. Open terminal and change directory to the folder after unzipping it
3. Run the command: **dotnet run <numNodes> <topology> <algorithm>**

For example: dotnet run 10 2D gossip

Here numNodes represents the desired number of peers in the network, and topology represents the type of topology to be used which can be one of full, 2D, line, imp3D and algorithm can have values of either "gossip" or "pushsum".

What is Working:

Our project successfully implements the Gossip and Push-Sum algorithms using the actor model in F#. We focused on accurate simulation and performance evaluation of these algorithms across different network topologies, including Full Network, 2D Grid, Line, and Imperfect 3D Grid.

Gossip Algorithm Implementation

Functionality: The Gossip algorithm for information propagation was fully implemented. Each actor efficiently selects a random neighbor and transmits the rumor, adhering to the specified termination condition.

Asynchronous Handling: Leveraging the Akka framework, we ensured fully asynchronous communication among actors, which is crucial for the Gossip algorithm's effectiveness.

Termination Condition: Actors successfully track the number of times a rumor is heard and cease transmission upon reaching the pre-set threshold, demonstrating correct algorithm behavior.

Push-Sum Algorithm Implementation

State Management: Each actor accurately maintains two quantities, s and w, and updates these values upon receiving messages.

Sum Estimation: The implementation correctly calculates the sum estimate at any given moment as s/w, aligning with the algorithm's requirements.

Termination Criteria: The actors appropriately terminate when the ratio s/w stabilizes within the specified threshold over three consecutive rounds.

Network Topologies
Diverse Topologies: All specified topologies (Full Network, 2D Grid, Line, and Imperfect 3D Grid) were successfully integrated into the simulation.

Topology Impact Analysis: The implementation allows for an effective analysis of the impact of each topology on the algorithms' convergence times and behaviors.

Performance and Convergence

Convergence Measurement: The system accurately measures and outputs the time taken to achieve convergence in each scenario.

Scalability and Efficiency: Our tests across various network sizes demonstrated the system's scalability and efficiency in handling large numbers of actors.

Dependencies:

```
<PackageReference Include="Akka" Version="1.4.26" />
<PackageReference Include="Akka.FSharp" Version="1.4.26" />
<PackageReference Include="Deedle" Version="2.4.3" />
```

Code Structure:

The code is in accordance with standards required to write an F# program. Comments have been placed above the code blocks to enhance the understanding.

	File Name	Functionality
1	Program.fs	This file is the entry point of the simulation, handling command line inputs to set up network topologies and initialize either the Gossip or Push Sum algorithm. It orchestrates the creation of nodes in the specified topology, manages their interactions, and tracks the performance and termination of the simulation.
2	Gossip.fsproj	This file is the project configuration file for an F# application using .NET, specifying the project's structure, dependencies, and target framework (net7.0). It includes references to various source files (like Util.fs, Simulator.fs) and external packages (such as Akka and Deedle), essential for the simulation.
3	PushSumNode.fs	This module defines the behavior of nodes in the Push Sum algorithm within a distributed system, managing state updates and message passing between nodes. It includes logic for initializing nodes, handling incoming messages, calculating sum estimates, and determining convergence based on a specified threshold.
4	Simulator.fs	This module defines the Simulator actor in the distributed system, responsible for monitoring the convergence of nodes and logging the results of either the Gossip or Push Sum algorithm. It handles termination messages from nodes, calculates convergence times, and outputs performance metrics to a CSV file.
5	Topology.fs	This file defines various network topology creation functions for a distributed system simulation, including line, 2D grid, imperfect 3D grid, and full network topologies. It maps each node to its neighbors based on the specified topology.
6	Util.fs	This module provides utility functions for the simulation, including rounding node numbers for grid topologies, selecting random elements or neighbors, and appending data to a CSV file. These utilities support the core functionality of the simulation across various network topologies.
7	FindNeighbour.fs	This file contains functions to determine the neighbors of a given node in either a 2D or 3D grid topology. The functions calculate neighbors based on the node's position in the grid and the overall size of the grid.
8	GossipNode.fs	This file defines the behavior of nodes participating in the Gossip algorithm, handling rumor spreading and reception. It includes logic for spreading a rumor to random neighbors, tracking the number of times a rumor is heard, and determining when to stop rumor transmission.
9	output.csv	This file stores the specific output or results generated during the simulation, which includes the Algorithm,Topology,Nodes,ConvergenceTime. This file is used to generate a Graph.

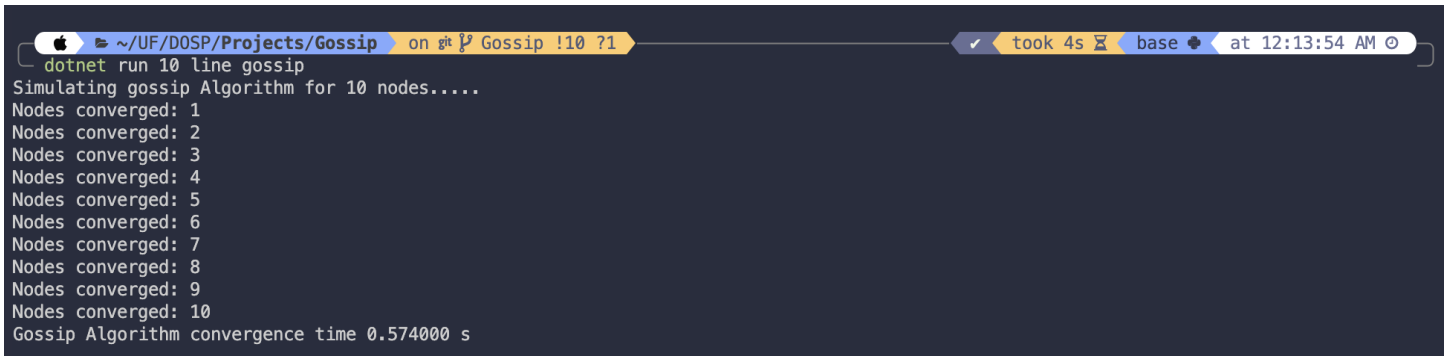
10	genGraph.py	This Python script automates the process of running the simulation across various network topologies, algorithms and NumNodes, capturing the output for analysis. It iterates through different node counts and topology-algorithm combinations, executing the simulation and collecting data for each configuration.

Output Analysis

The program outputs the convergence time for the selected algorithms and also prints how many nodes have converged. For the Push Sum algorithm it also outputs the s/w value for each Actor Node.

Screenshots

Below is a sample screenshot of the program output for > dotnet run 25 line gossip.



dotnet run 10 2D possum

Largest network

In our largest network tests, the Gossip algorithm with 10,000 nodes showed the Full Network topology as the most efficient, converging in 2.749 seconds, followed by the 2D Grid and Imperfect 3D Grid. For the Push Sum algorithm in a 5,000-node network, the Full Network again excelled with a convergence time of 4.215 seconds. These results highlight the Full Network's superior performance in large-scale networks.

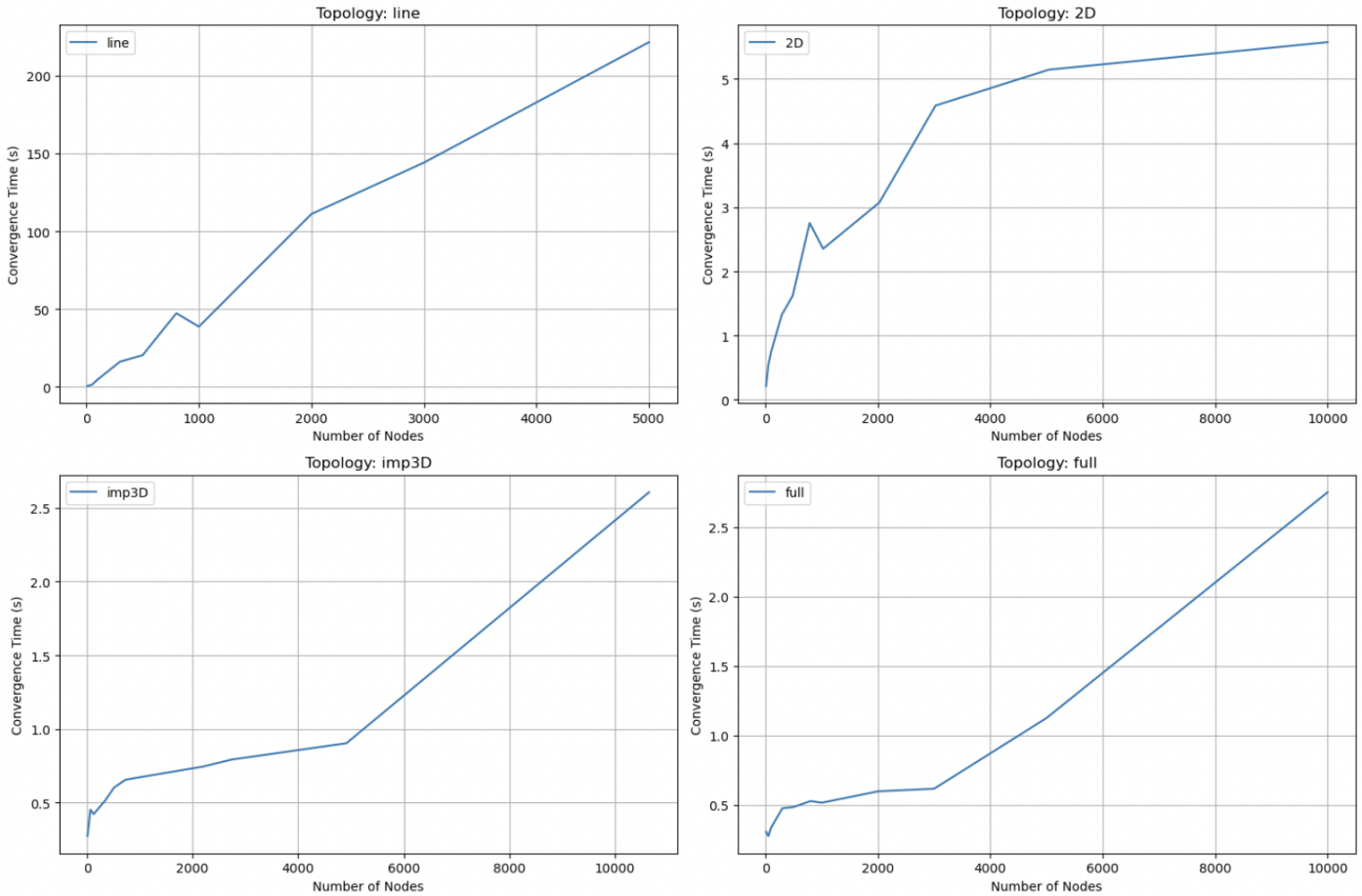
Convergence Time Results

Based on the various simulations that we ran nf the Gossip and Push Sum algorithms across various network topologies and node counts, the following trends and observations can be highlighted:

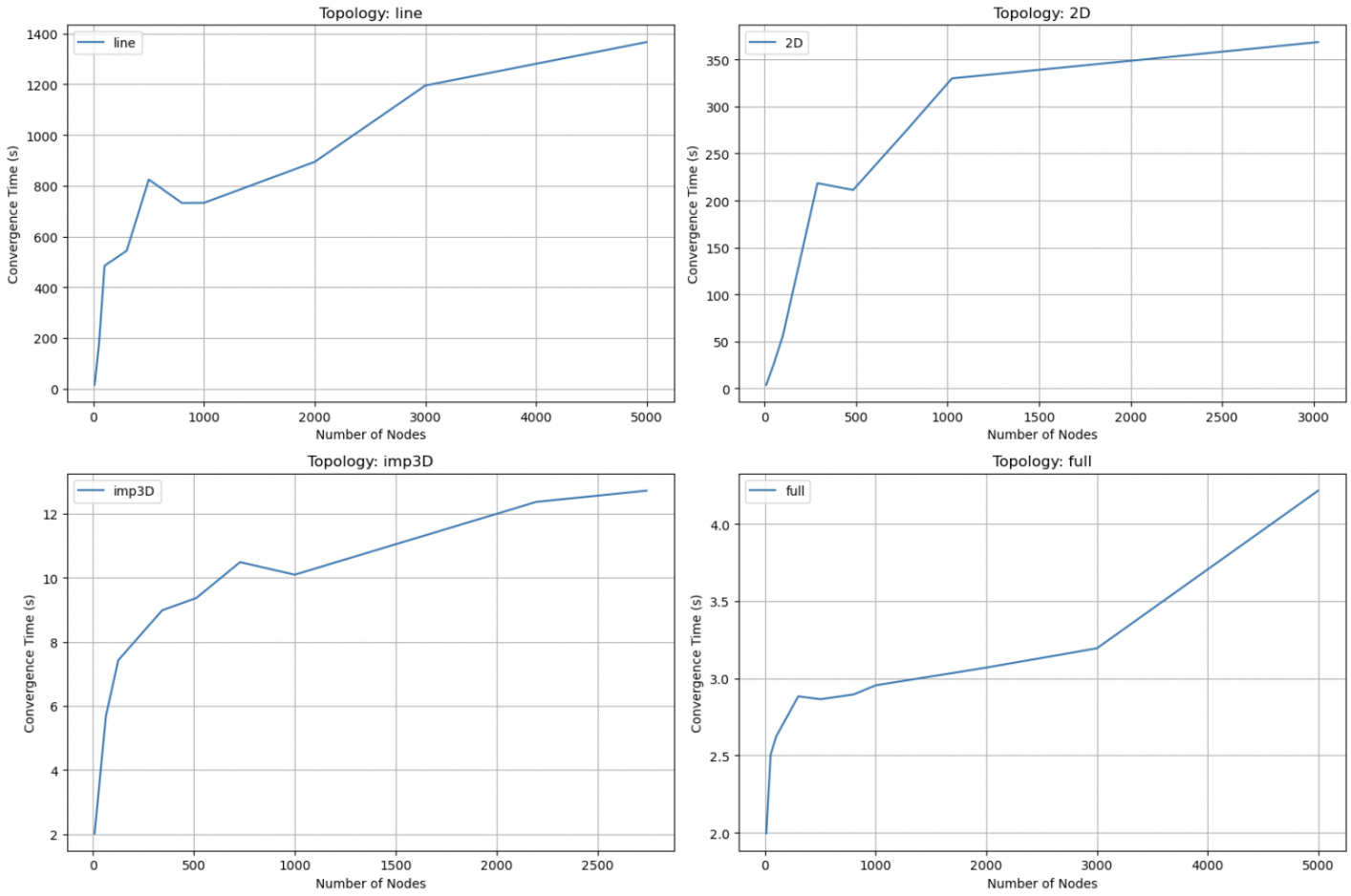
1. The Gossip algorithm consistently shows faster convergence times compared to the Push Sum algorithm across all topologies.
2. The Full Network topology yields the quickest convergence times for both algorithms, likely due to the direct connectivity between all nodes.
3. As the number of nodes increases, there is a marked increase in convergence time for both algorithms. This trend is more pronounced in the Push Sum algorithm, indicating its sensitivity to network size.
4. The Gossip algorithm scales better with the increasing number of nodes, especially in the Full Network and Imperfect 3D Grid topologies.
5. In the Line topology, both algorithms exhibit a significant increase in convergence time as the node count grows, with the Push Sum algorithm being particularly affected. This indicates that the Line topology is less efficient for information propagation and sum calculation.
6. The 2D Grid and Imperfect 3D Grid topologies offer a balance between the extremes of the Line and Full Network topologies, with moderate convergence times for both algorithms.

Results Visualization:

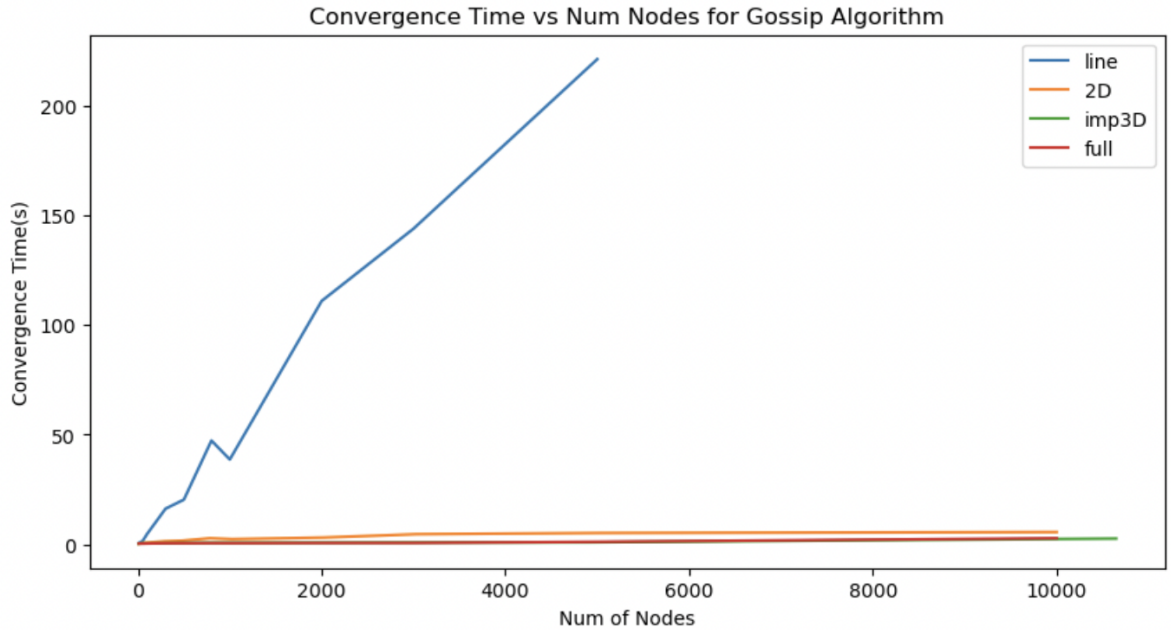
1. **Convergence Time vs NumNodes for all Topologies in Gossip Algorithm**

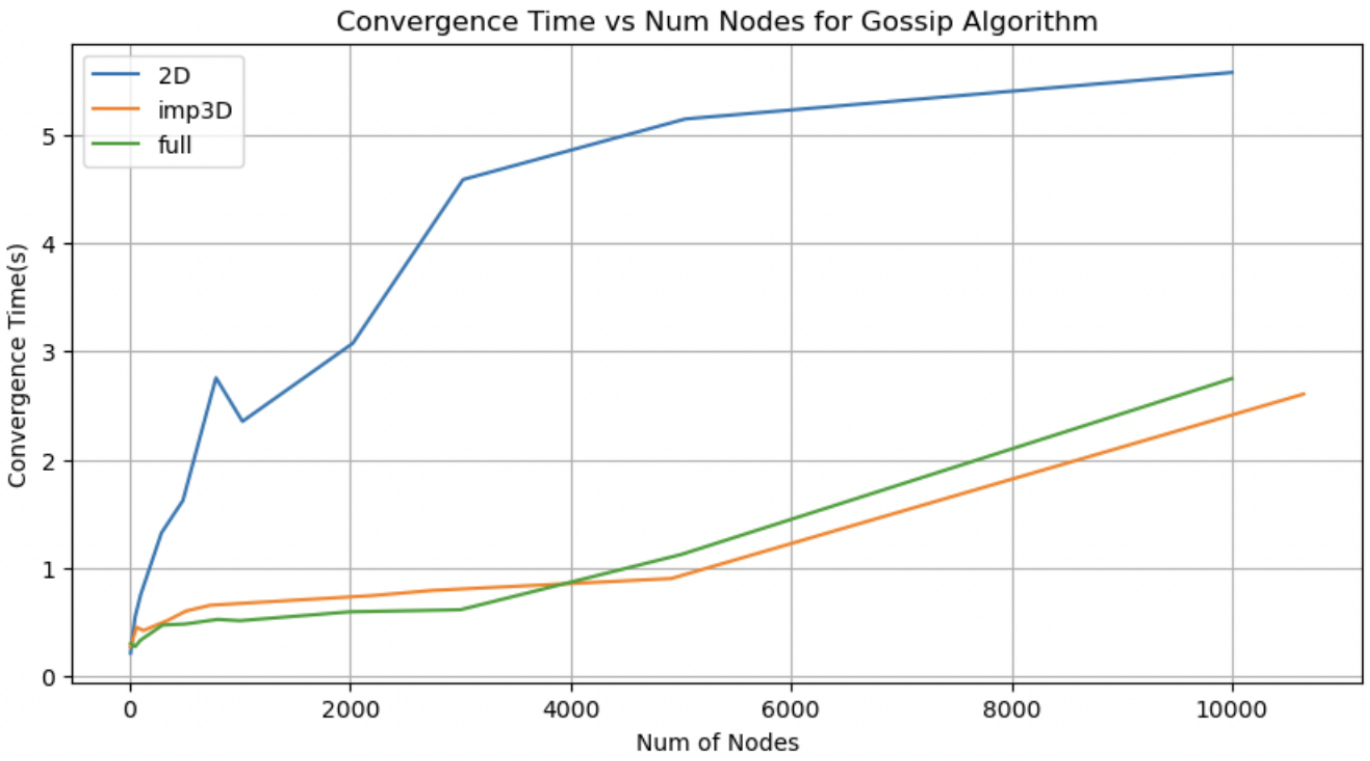


2. Convergence Time vs NumNodes for all Topologies in Push Sum Algorithm

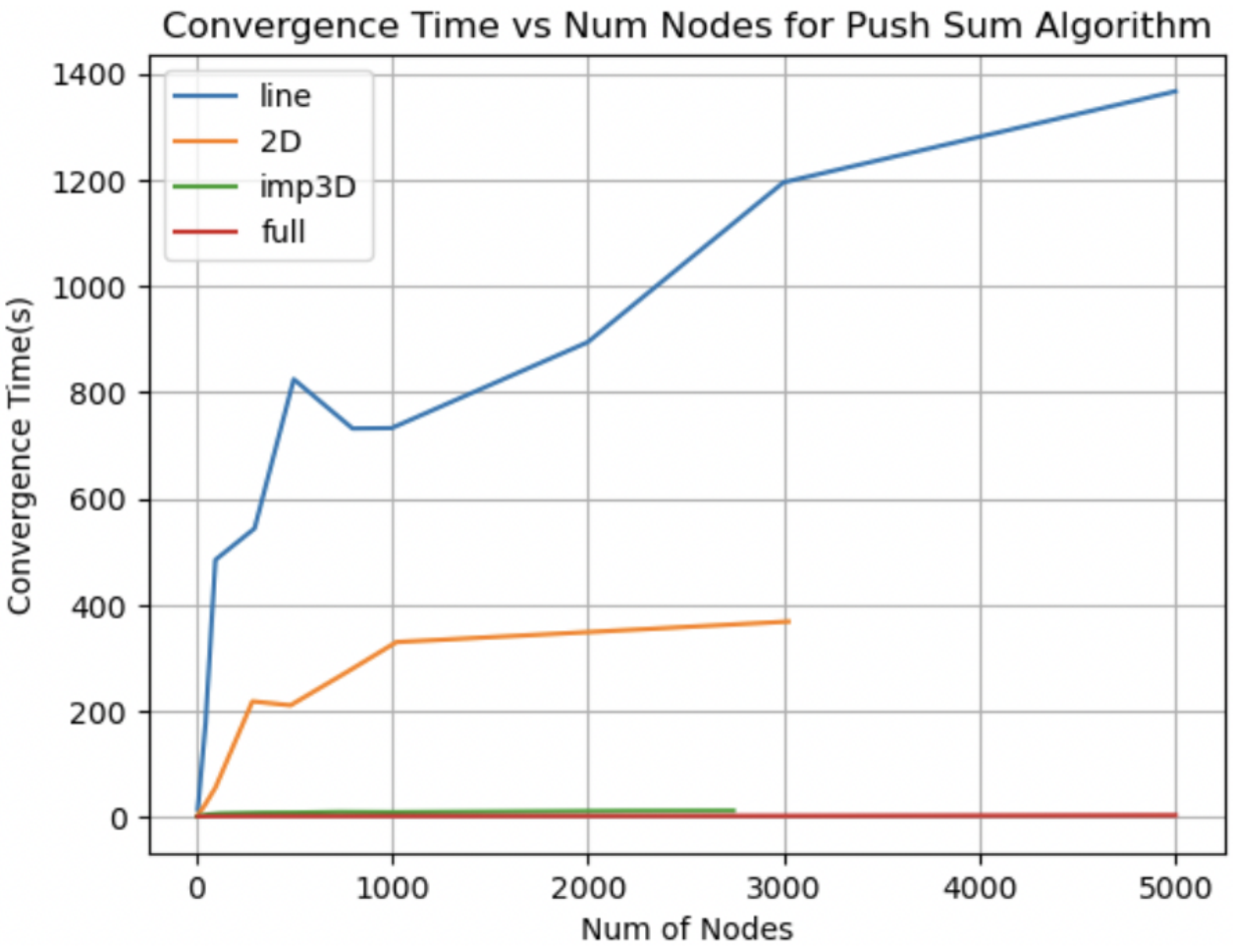


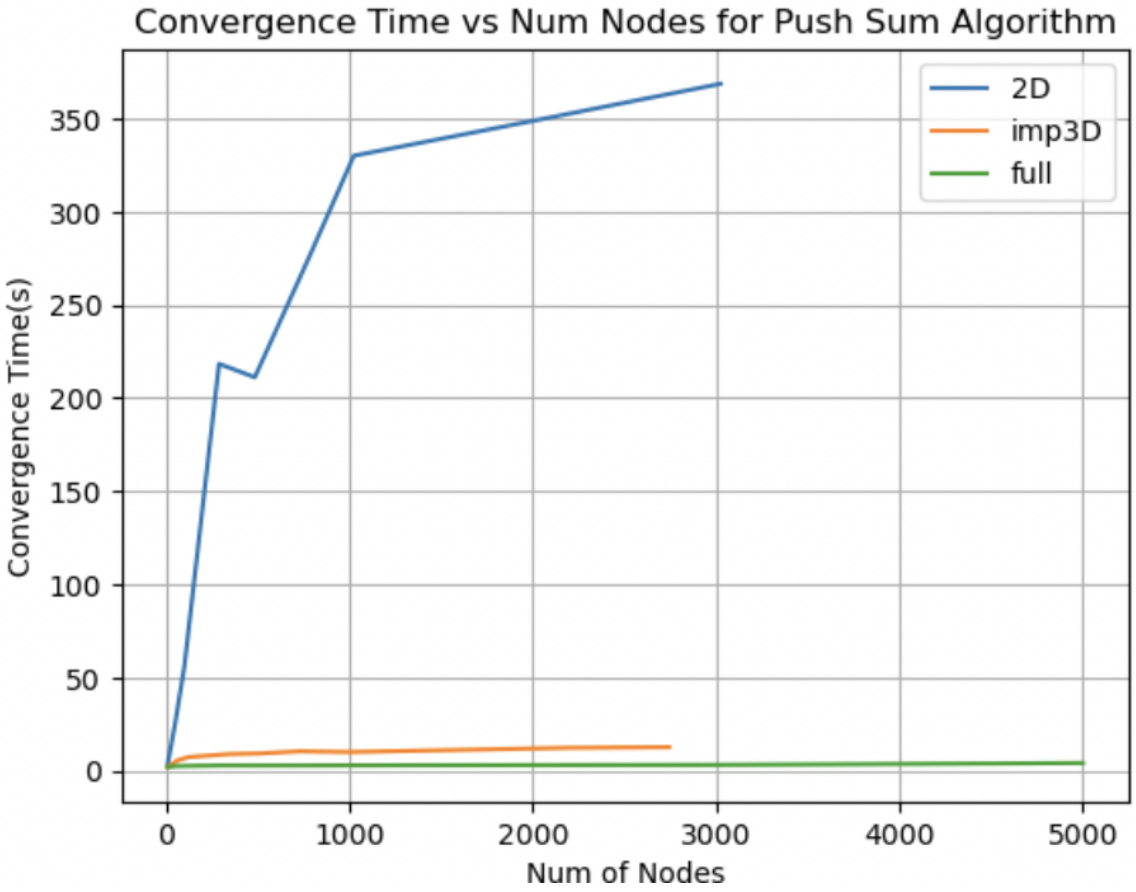
3. Comparison of Various Topologies in Gossip Algorithm

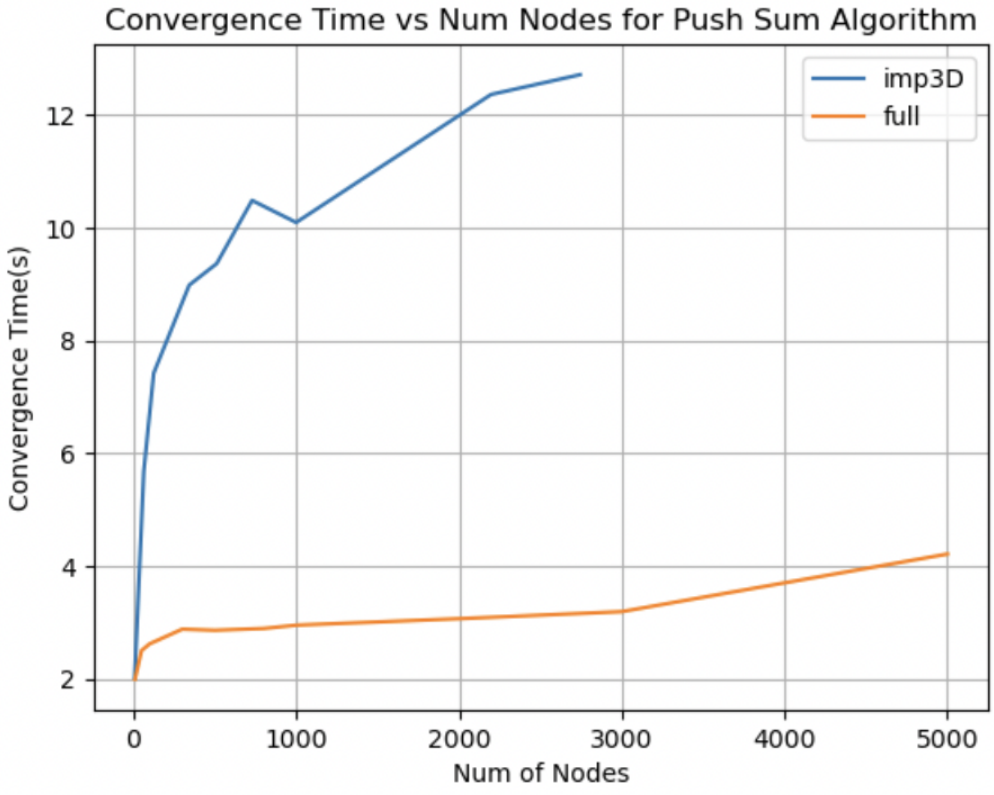




4. Comparison of Various Topologies in Push Sum Algorithm




Convergence Time vs Num Nodes for Push Sum Algorithm
2D
imp3D
full
Convergence Time(s)
0
50
100
150
200
250
300
350
0
1000
2000
3000
4000
5000
Num of Nodes


Convergence Time vs Num Nodes for Push Sum Algorithm
imp3D
full
Convergence Time(s)
2
4
6
8
10
12
0
1000
2000
3000
4000
5000
Num of Nodes

Assumptions for the Simulation

Network Stability and Reliability

Constant Network Topology: It is assumed that the network topology (Full, Line, 2D Grid, Imperfect 3D Grid) remains constant throughout the simulation. No nodes are added or removed, and the connections between nodes do not change.

Reliable Communication: All messages sent between nodes are assumed to be reliably delivered without loss, corruption, or reordering.

Node Behavior and Capabilities

Homogeneity of Nodes: Each node in the network is assumed to have uniform capabilities in terms of processing power, memory, and network bandwidth.

Synchronous Operation: Nodes are assumed to operate in a synchronous manner, with a uniform delay for message processing and transmission.

Conclusion

In this project, we effectively implemented and analyzed the Gossip and Push Sum algorithms in distinct network topologies using F# and Akka. The findings reveal significant differences in the algorithms' performance across topologies like Full Network, Line, 2D Grid, and Imperfect 3D Grid. The Gossip algorithm proved faster in spreading information, while Push Sum was more efficient in calculating network-wide sums. These results underscore the critical influence of network structure on distributed algorithm efficiency. The project not only deepened our understanding of these algorithms but also demonstrated the practical capabilities of the Akka framework in a distributed setting.

Contributions:

Girish Vinayak Salunke
Engaged in in-depth research and attended tutorials to understand the Push Sum Algorithm.
Implemented the network topologies (Full Network, Line, 2D Grid, Imperfect 3D Grid) for the simulation, focusing on the structural nuances and impacts on the algorithms, and documented the development process.

Janhavi Shriram Athalye
Handled the setup of the Akka framework environment and managed dependencies in the F# project.
Developed and implemented the core logic for the Gossip algorithm, ensuring efficient message propagation across various topologies, and contributed to the project's comprehensive documentation.

Reema Solan
Conducted rigorous performance evaluation of the algorithms, analyzing convergence times and efficiency, and documented these findings and plotting graphs.

Srujith Reddy Narra
Led the debugging and optimization efforts, improving code efficiency and resolving any runtime issues.