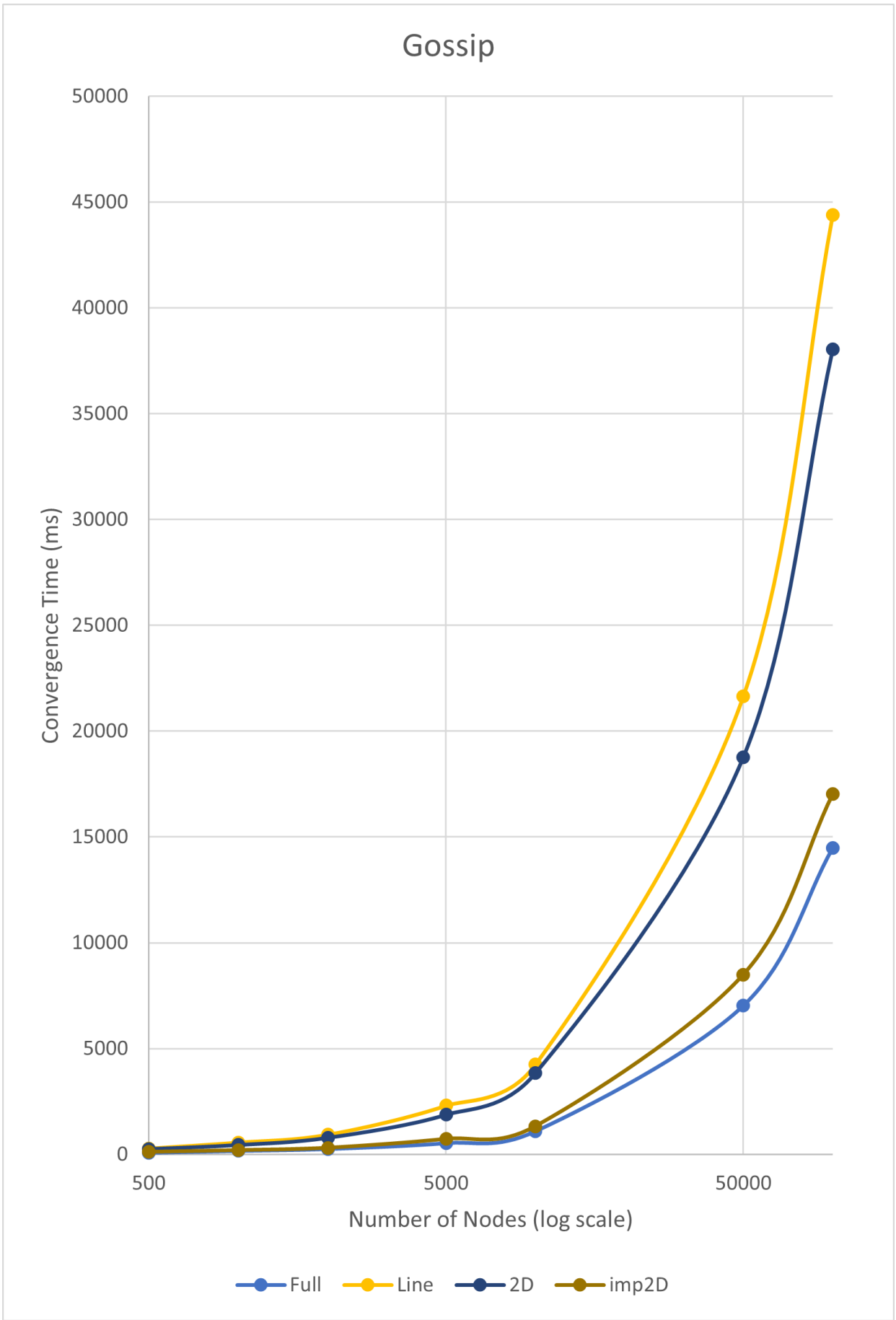
Project 2 – Gossip Simulator

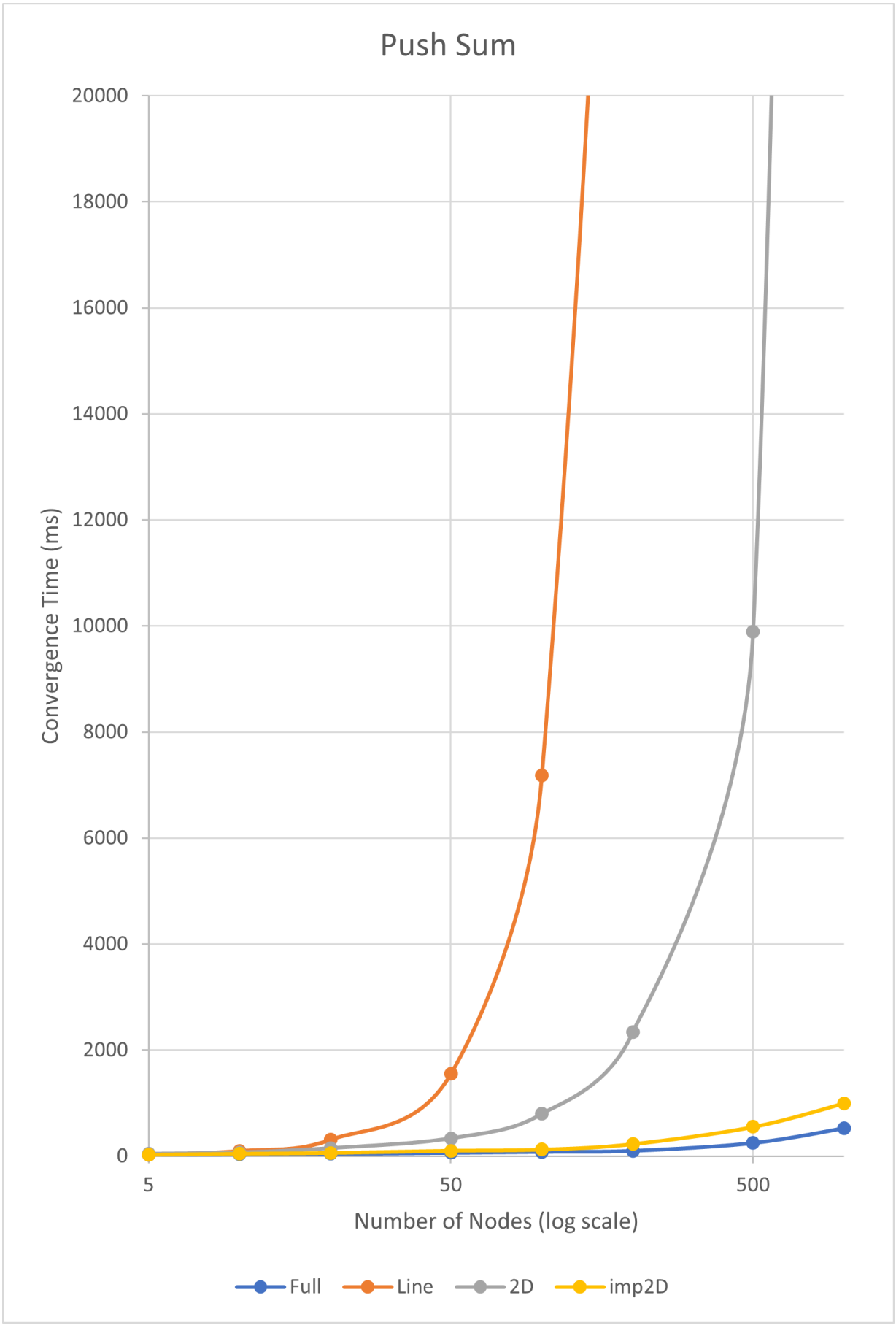
Report

The graphs for comparison of topologies are given on the next 2 pages. The general trend is that lower number of nodes and higher freedom of choosing a neighbor directly relate to lower convergence time in gossip.

This is clearly seen in the graph for gossip where full network takes the least time to converge, having the most freedom of choosing a neighbor, followed by imp2D where there are slight restrictions. Line and 2D perform the same as they are quite restricted, and the initial nodes stop transmitting early on as the message bounces between the initial few nodes quite a lot. This obviously also means that full and imp2D allows the message to reach much more nodes in the same time than line or 2D.

In push sum the trend is similar and the line and 2D take much more time to converge and their gap is even more wide than in gossip. In push sum the ratio converges to the average value and it gets closer to the real average the later a node converges so the nodes that converge at the very end will have average very close to the real one.


Gossip
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
Convergence Time (ms)
500
5000
50000
Number of Nodes (log scale)
Full
Line
2D
imp2D


Push Sum
20000
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
Convergence Time (ms)
5
50
500
Number of Nodes (log scale)
Full
Line
2D
imp2D

The system consists of N actors where N is the number of nodes. Each actor is identical and acts as a node. Initially the first message is passed to 1 actor and then the actors follow the logic to send the message to a random neighbor when they receive a message until they converge. This allows the message to be passed to the whole network.
The convergence criteria for any node in gossip is when it has received the rumor 10 times. In push sum it is when the ratio changes less than 10^-10 in 3 consecutive rounds. The execution stops when except 1 node, all others are converged, and this is because if a single node is left it cannot converge on its own since sending messages to itself is not allowed.

All converging nodes are tracked and when sending the message, it is ensured that the neighbor selected for sending the message is not already converged as it will not send the message further again thus stopping the message passing. So, a neighbor is randomly selected and checked if its converged or not. If it is not converged, message is passed to it. Otherwise another neighbor is randomly selected and this can happen up to 500 times and if still there is no non-converged neighbor then any non-converged node will be sent the message regardless of topology constraints and this is just to make sure all nodes do converge but the number is set to 500 so that this only happens in cases where really no valid neighbor exists.