Project 2 Report

In this project we implemented the gossip algorithm. Two utilities of this algorithm are being realized, message passing and push sum.

The core functions of our application are two functions, named respectively of their utility

```
gossip(GridType, MasterNode, Index, Nodes, ActualMessage, RecievedMessageCount, RandAccessDimIndex, RandNodeIndex) ->
    TerminationCount = 10,
    if
        RecievedMessageCount < TerminationCount ->
            receive
                {gossip, Message} ->
                    lists:nth(getRandomNeighbour(GridType, Index, Nodes, RandAccessDimIndex, RandNodeIndex), Nodes) ! {gossip, Message},
                    if
                        RecievedMessageCount > 0 ->
                            {master, MasterNode} ! {finito};
                        true ->
                            ok
                    end,
                    gossip(GridType, MasterNode, Index, Nodes, Message, RecievedMessageCount + 1, RandAccessDimIndex, RandNodeIndex)
                after 50 ->
                    if
                        RecievedMessageCount > 0 ->
                            lists:nth(getRandomNeighbour(GridType, Index, Nodes, RandAccessDimIndex, RandNodeIndex), Nodes) ! {gossip, ActualMessage},
                            gossip(GridType, MasterNode, Index, Nodes, ActualMessage, RecievedMessageCount, RandAccessDimIndex, RandNodeIndex);
                        true ->
                            gossip(GridType, MasterNode, Index, Nodes, ActualMessage, RecievedMessageCount, RandAccessDimIndex, RandNodeIndex)
                    end
            end;
        true ->
            ok
    end.
```

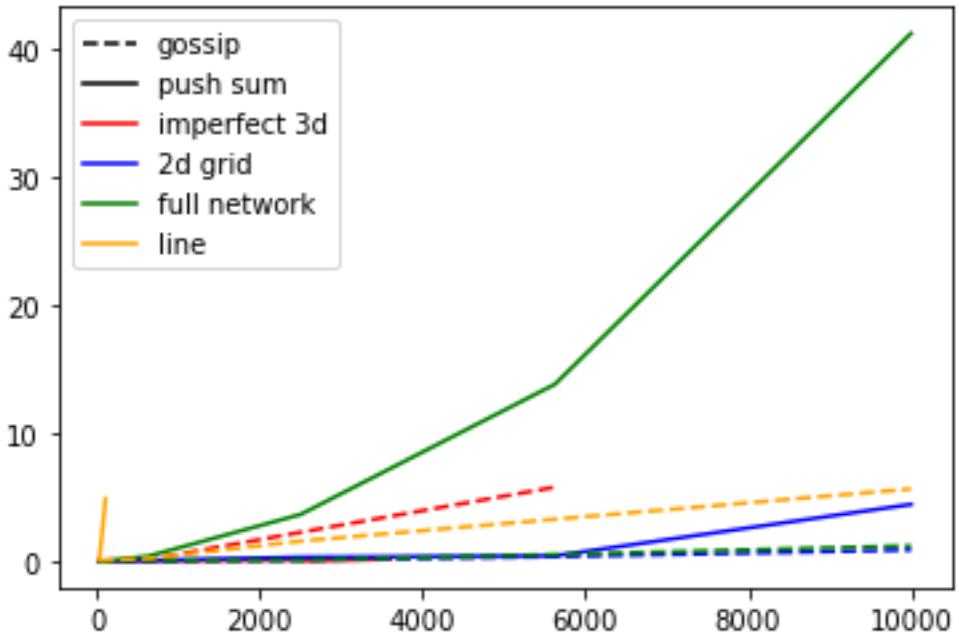
Gossip function is run by each working process. Once started, it waits for the master node to send out the first initializing message to a random process via:

```
AlgorithmType == "Gossip" ->
    lists:nth(getRandomNumber(1, length(Nodes)), Nodes) ! {gossip, "Advanced message"},
    bossWaitForFinish(gossip, length(Nodes), GridType, StartTime, NumberOfNodes);
AlgorithmType == "PushSum" ->
    lists:nth(getRandomNumber(1, length(Nodes)), Nodes) ! {pushSum, 0, 0},
    bossWaitForFinish(pushSum, length(Nodes), GridType, StartTime, Nodes)
```

Every gossip process, upon receiving a message, will pass along the message to a different node depending on the grid topology used. Each process keeps a counter for the maximum times of message passing.

We implemented 4 different topologies, imperfect 3d grid, 2d grid, line, and full network.

We then plotted run time for each topology against its runtime.



Several observations were made from the above graph.

1. Line topology is suitable for gossip but proven to be vary incapable for push sum algorithm. We hypothesize that due to the relative in accessibility between distance nodes, it is very difficult for two nodes from the opposite ends of the line to converge. This is worsened by the fact that we chose the direction of the message passing randomly.
2. Full network has proven to be very useful for gossip but not push sum. This result is somewhat surprising to us, since full network is the opposite of line topology. Both topologies are proven to be bad for push sum algorithm but suitable for gossip. Our hypothesis is that due the random nature of this topology, some nodes would take a very long time to converge due to pure chance.
3. 2d grid performs the best in both scenarios. We think that 2d grid is the best combination of randomness and order.