Shay Smith Report

Source: DivyanshuSaxena/Distributed-GHS: Implementation for computing the Minimum Spanning Tree using GHS Algorithm in a distributed fashion (github.com).

How correctness was verified:
The weight of the MST generated by the GHS algorithm is compared to the weight of an independently running Kruskal's algorithm, and it is asserted that the two are equal. Kruskal's algorithm is possible from the main.py file because it, unlike each independent node, has complete knowledge of the entire system.

GHS paper theoretical analysis of algorithm's communication cost and processing time:

- Worst case total messages sent is O(2E + 5NlogN). Each message consists of at most one edge weight, one integer between zero and log2N, and three additional bits.
- Worst case time complexity is O(NlogN).

How I evaluated the algorithm's communication cost and processing time:
For evaluating communication cost, the total number of messages to generate the MST was recorded and plotted with other runs of the program that used different input graphs. For evaluating time cost, the total time to generate the MST was recorded and plotted in a similar fashion.

From these plots, we can see the relationship between nodes vs. total messages, edges vs. total messages, nodes vs. total time, and edges vs. total time

What dataset I used and why I chose it:
I didn't end up using a SNAP dataset to evaluate the algorithm. I downloaded a couple, but the formatting wasn't clear to me and was not aligned to what the GHS program expected as input. Rather, I used a graph creation script to create randomly connected graphs for use as test cases. Using this script, I generated graphs with 20 to 340 nodes.

I was unable to run the program on a graph of more than 340 nodes. With 360 nodes, almost 10GB of RAM was allocated and nearly 40,000 threads were spawned (used "htop" to get memory and thread statistics). I'm not sure how the python multiprocessing module works internally, but it was too intensive for my laptop, especially when each node is given its own memory space.

<u>Results:</u>

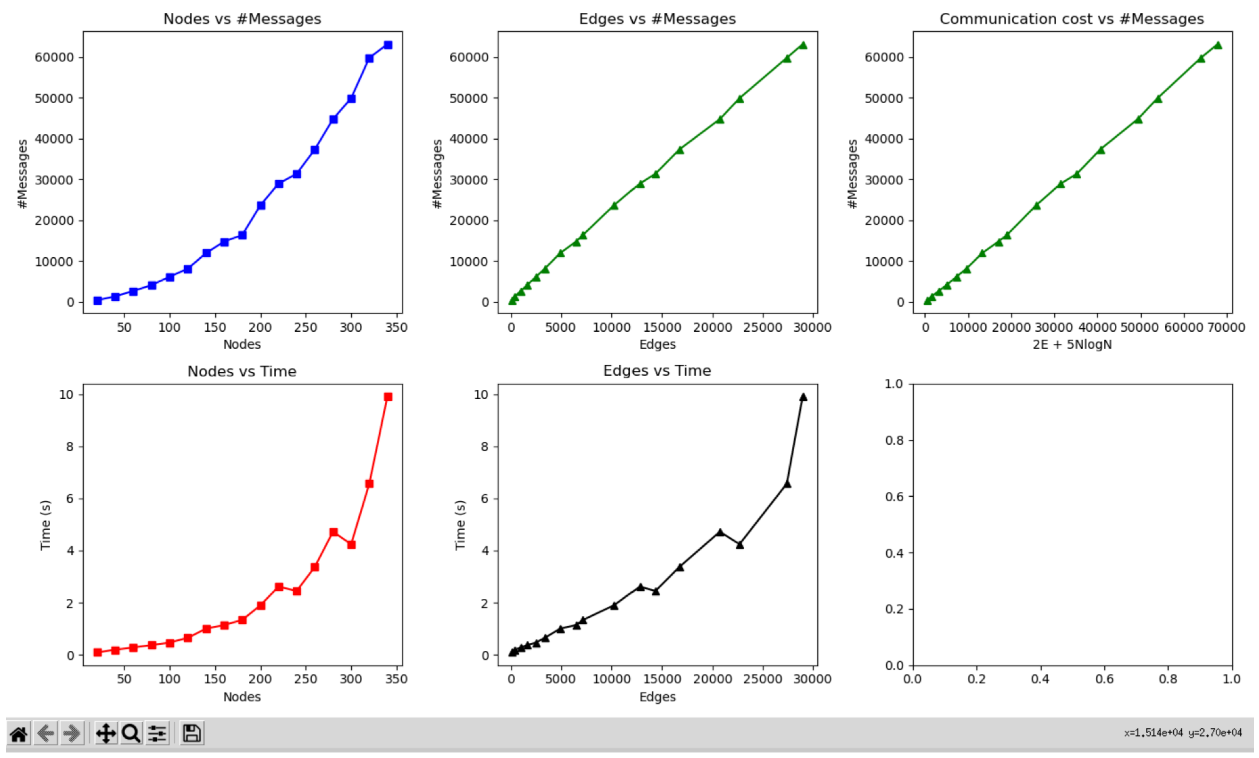

Figure 1: Performance plots for randomly connected graphs

<u>My analysis of results (+ do they line up with the theoretical analysis?):</u>
I believe my results line up closely with the theoretical analysis. Let's examine Figure 1. In the first plot, we see a super-linear polynomial relationship (~NlogN) between nodes and messages sent. In the second plot, we see a linear relationship between edges and messages sent. Both of these plots back up the idea that the communication cost is O(2E + 5NlogN). To further check this, a third plot was created to show the relationship between 2E + 5NlogN and messages sent. It is heavily dominated by the edges though, as the ratio between nodes and edges was not large enough for the 5NlogN element to be significant. For example, the largest graph evaluated had 340 nodes and 28965 edges, which, when substituted into the communication cost function is (2*28965 + 5*340*log(340)) = (57930 + 4303). Ratio: (4303/57930) = 7.5%.

From the timing plots, it is clear that the relationships between time and nodes/edges is not linear. The worst case complexity is supposedly O(NlogN), but the timing plots look relatively exponential. That may be because of system restraints, however.

It would definitely be interesting to see the results with larger input graphs, but I believe the results I've displayed here still display strong evidence to support the theoretical analysis.