Testing on large data

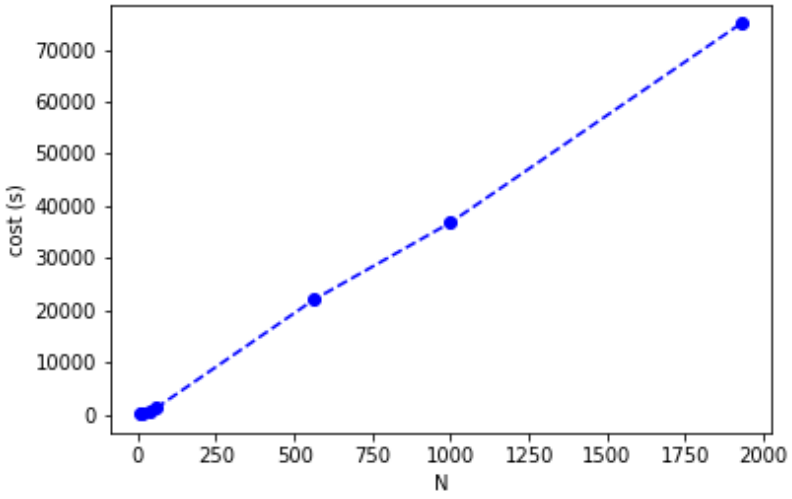
Going forward we want to see how our solution scales on larger inputs.

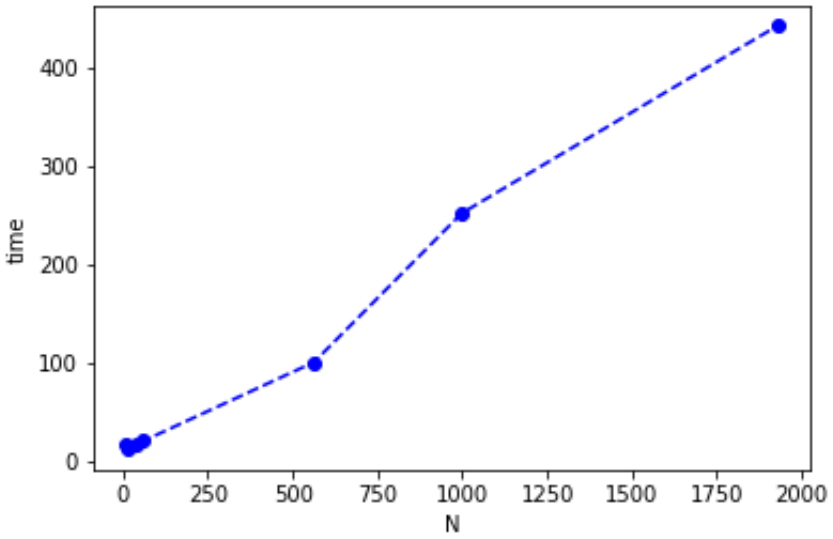
We ran ACO algorithm for 750 nodes, but unfortunately one iteration of the algorithm took more than 20 minutes so we won't go further testing for larger inputs, concluding that an ant colony approach won't scale well with a number of nodes bigger than 500.

The GA algorithm, on the other hand, offers an optimistic view of how it behaves on large data. Again, we run 10 iterations for each algorithm and computed the minimum, maximum, average and standard deviation of the cost with the additional information about the time it took in seconds:

Data	Minimum	Maximum	Average	Stddev	Avg. Time (s)
N=563	21144	22566	22035.7	461.1	101.98
N=998	35556	38877	36767.7	1053.5	252.19
N=1927	72848	77434	74977	1919	442.66

To make a clear picture about this scaling, looking at the following graphs we can see that the GA solution scales laniary with the input both on cost and time.





A note is that we were unable to find comparisons for inputs of such a large value, the biggest input would fall under 400 nodes. As a result we were unable to compare it to the state-of-the-art approach.

Conclusions

As the input grows by the ACO approach doesn't offer much practical value because it would take too much time to be feasible. What it is to remind here is that this approach offers a better cost than the other approach so we would recommend this for smaller inputs.

Even if the genetic algorithm was left behind on small inputs in terms of finding a minimum cost, it shows its value by being linearly scalable with the input size. This would be the preferred way if we are up to teal with large graphs.