COMP26120 Lab 10

Wenchang Liu

March 5, 2019

1 The small-world hypothesis

Statement and how to test it:

I think the small-world hypothesis is false, and I will test for it by checking:

For each 2 people in the graph, try to find a shortest path whose length is less than (or equal to) 6. If each pair of people has a path that is not longer than 6, then the graph is a "small world", otherwise, the graph is not a "small world".

2 Complexity Arguments

Compare time complexity of Dijkstra and Floyd algorithm:

Dijkstra using priority queue has the complexity of $O\left((|E|+|V|)\,log|V|\right)$, where E is the number of edges, and V is the number of vertexes. We also need to apply Dijkstra for every pair of vertexes, which means that the overall complexity would be $O\left(|V|*((|E|+|V|)\,log|V|)\right)$.

Floyd has the complexity of $O\left(|V|^3\right)$, where V is the number of vertexes.

And we can approximately calculate the scale of time for Caltech.gx and Oklahoma.gx using Dijkstra and Floyd:

Dijkstra: $O\left(|V|*((|E|+|V|)\,log|V|)\right)$

(1) Caltech.gx: $769*((33312+769)*log\,(769))=251,254,668$

(2) Oklahoma.gx: $17425*((1785056+17425)*log\,(17425))=442,506,552,000$

Floyd: $O\left(|V|^3\right)$

(1) Caltech.gx: $769^3=454,756,609$

(2) Oklahoma.gx: $17425^3=5,290,763,640,625$

We can see from the approxiamation calculated using time complexity, Dijkstra algorithm is much less than Floyd algorithm, so Dijkstra algorithm is more efficient.

3 Part 2 results

Part2 experiment:

Part2 implemented Dijkstra algorithm using priority queue. Using priority queue, we can easily remove the node with the minimum distance from the priority queue, when we remove a node from priority queue, it means that we add the node into the Dijkstra cloud and we have already calculated the shortest path to this node.

Statistics of Caltech.gx and Oklahoma.gx:

Caltech: Time: 0.43s,
Average unreachable: 13.91,
Average larger than six: 0,
Average distance: 2.32,
Conclusion: it is not a "small world".

Oklahoma: Time: 2583.18s,
Average unreachable: 10,
Average larger than six: 0.19,
Average distance: 2.77,
Conclusion: it is not a "small world".

Compare Caltech.gx and Oklahoma.gx:

According to the time scale we calculated in part1, time scale of Oklahoma.gx is $442,506,552,000/251,254,668 = 1761.19$ times larger than Caltech.gx, and according to the execution time we recorded, Oklahoma is $2583.18/0.43 = 6007.4$ times longer than Caltech.gx, which is considered to be agreed with the time complexity if we consider some small issues like different execution environment(server cpu), constant factor, initialize arrays to zeros, etc.

4 Part 3 results

Part3 experiment:

Using heuristic algorithm(find the node with the largest outdegree as the next step) to save some time on graphs with large scale, and compare the statistics with Dijkstra algorithm.

Statistics of Caltech.gx and Oklahoma.gx:

Caltech:
Time: 0.38s,
Average unreachable: 36.81,
Average larger than six: 208.93,
Average distance: 15.50,
Conclusion: it is not a "small world".

Oklahoma:
Time: 497.64s,
Average unreachable: 3164.17,
Average larger than six: 8766.88,
Average distance: 67.67,
Conclusion: it is not a "small world".

5 Conclusions

Conclusions of experiments of part2 and part3:

Part2:

Both Caltech.gx and Oklahoma.gx are not strictly "small world", the graphs have some "isolated" people that cannot be reached. Apart from that, there are not much distance more than six shortest paths in these two graphs. So in my opinion, these two graphs are close to "small world" but they are not.

How close to "small world" according Dijkstra (matching rate: "100%" regarded as strictly matching "small world"):

Formula: $(numberOfAllNodes-(unreachableLargerThanSix))/numberOfAllNodes$

Caltech.gx: $(769-(13.91+0))/769 = 98.2\%$

Oklahoma.gx: $(17425-(10+0.19))/17425 = 99.9\%$

Part3:

Compared with part2, the number of average unreachable nodes and average shortest distance between nodes and the number of distance larger than six nodes has increased a lot, the time complexity and running for this algorithm is lower, however, the actual results we get have limited accuracy. We have so many nodes that is miscalculated as unreachable, not to mention distance larger than six nodes, especially when we implement this on a graph with large scale.

To what extent it works:

We can evaluate the algorithm compared with Dijkstra (miscalculated rate: "0" regarded as perfect):

Formula: $((heuristicUnreachable+heuristicLargerThanSix))/numberOfAllNodes$

Caltech: $((36.81+208.93)-(13.91+0))/769 = 30.1\%$

Oklahoma: $((3164.17+8766.88)-(10+0.19))/17425 = 68.4\%$

So we can see that the algorithm becomes worse when the scale of graph becomes larger.