Name: Ryan Goh

Class: 4K

**What did you find out? learnt? [not more than 2 pages]**

I found out that the exact solutions to some problems with an acceptable runtime for real world tasks may be difficult, or maybe even impossible to find. These problems are known as NP-hard problems, (or NP-complete for the decision variant of the problem), and as for now has no polynomial solution that exists. However, there would be various kind of approximate algorithms out there that can help to “solve” the problem with reasonable and acceptable output, while taking a much shorter time. These heuristic algorithms essentially traded accuracy for time. This is supported through the case study that I had closely examined: the Travelling Salesman Problem. The travelling salesman problem seeks to find a minimum route that visits all the nodes defined in a 2D space, and return to the starting point. From what I know, the best algorithm to find an exact answer for the travelling salesman problem is dynamic programming with a time complexity of (N^2\*2^N) where N is the number of nodes. The other solution is brute force with O(N\*N!). Both are impractical for real world instances. Fortunately, there are some heuristic algorithms that can solve this problem, namely nearest neighbour algorithm, where we go to the nearest node that we have not visited from the current node and “shortest edge construction” algorithm, where we pick the shortest edges to form a valid path. A valid path is formed when the graph is connected, does not have a cycle with less than N edges, and does not have a node with more than 2 edges connected to it. The former takes O(N^2) time but is 25% worse than optimal; while the latter takes O(N^2logN) but is 20% worse than optimal. Moreover, after a valid construction of a path using a constructive heuristic algorithm mentioned above, I also found out that we can tweak the route using other algorithms to get a shorter path! One of these is the k-opt algorithm, where we repeatedly use an operation: remove any k edges and re-join them to get a shorter path. This can be done as long as it is possible to do so, ensuring 5% worse for 2-opt and 3% worse for 3-opt. Of course, the larger the k, the worse the runtime; as we need to consider more ways to connect the graph with larger k. In fact, the number of possible operations we can do to get a shorter path can be exponential too! Fortunately, average number of operations is O(N) if we use a greedy algorithm. Hence, for 2-opt, assuming that the complexity of joining the nodes is O(N) (O(log N) is possible with a splay tree), the average time complexity is O(N^2+N^2) = O(N^2), and worst time complexity is O(cN + N^2) where c is the number of operations that is required. With these examples, and testing with these algorithms by coding the solutions, I also found out that even for heuristic algorithms, a better solution may also require more time. As such, I realised why in the real world, people often contemplate very hard on the algorithms they choose to solve a problem – whether they want to prioritise time, accuracy or even both?

**What is the “roadmap” you used to find out what you need? [if someone else is interested in this area, how would you advise them? to be more efficient etc]**

The roadmap is surprising simple for me. I used Google to search for the travelling salesman problem and the Wikipedia page listed all the heuristic algorithms that I can use to solve the problem! However, I still decided to find a research paper (fortunately there is already one since this problem is actually very well researched on) that lists down the accuracy and time complexity also, to cross reference with what I learnt. (In fact, I was once misled to think that the 2-opt algorithm is at **worst** O(N^2), however I found on another research paper that there may be some cases where the number of operations that can be done is more than O(N^2), though those cases are rare, and average is indeed O(N^2).) For me, I also coded out programs to test the effectiveness and efficiency of those algorithms, to prove what I had found out. That was when I realised that some algorithms may require other sub-algorithms (such as the usage of a union find disjoint set for the “shortest edge construction” algorithm), which I would have to learn if I do not know. I obviously would not realise that such details needed to be learnt if I did not try implementing it!

For any person who would like to find out about heuristic algorithms, there is no other way other than to search about the problem, cross reference, and try out on your own, researching more if necessary.

**Reflection on your findings [stumbling block, pitfalls etc] [not more than 1 page]**

By doing the slides for my presentation, I learnt to choose content from my findings that I can share with others without making them too confused. For example, I also learnt about the algorithm where one finds the minimum spanning tree of the nodes, then duplicate the edges to find a Eulerian path, then construct the TSP tour from there. However, explaining these 2 concepts is difficult and unrelated to the crux of my sharing – the efficiency of heuristic algorithms – at all! Fortunately, I stopped myself from going down that route.

**Reflection on your sharing [not more than 1 page]**

From making the video, I found it the most difficult to squeeze all my content into 4 minutes without making it too fast. To do so, I summarized all my points, and put more focus on the part where I learnt (the heuristic algorithm part), where I described the algorithm in more detail than the exact algorithm. With some video editing (no speeding up), I managed to fit everything, including a run through of my code into 4 minutes. Here, I learnt that my sharing should be focused on the main points, with the time limit helping me to eliminate some unnecessarily complicated explanations (such as how the dynamic programming algorithm works) that I may had done without it.

**Comments on Cluster [how your cluster member help you?]**

I did not get much help from my cluster, because I started the project quite late (but I ensured myself that I still have enough time to research about the topic), due to some commitments that I had during my June Holidays. So, when I tried to get some feedback from my cluster members, they were already too busy to help me comment on my draft video. Hence, I do not have any comments from my cluster.

As such, I learnt that I should have reached out for my cluster member a little earlier, or set up some deadlines for my project such that I can finish earlier, and have some time left for my cluster members to help me feedback.

**Resources that guide you [most important ones on top in bold, could be links for articles, books, video]**

(Link of most important resource is in **bold**.)

I first found out that heuristic algorithms existed by reading through some notes taught in IOI Training (I have the notes and glanced through them when I have some interest – but of course I don’t understand much things at that point in time) The slides are here at <https://docs.google.com/presentation/d/1FUEQFUlIDDb6HEy2dAu3K-kuuCBYBqKe3ZPhQDYfHP4/edit#slide=id.g3e3cdb4ffd_0_439>.

Next, I found out more in the Wikipedia page for Travelling Salesman Problem. (<https://en.wikipedia.org/wiki/Travelling_salesman_problem>), however the description for the 2-opt algorithm is a little vague, so I explored other sources and eventually found a research paper about it that explains the heuristic algorithms in much more detail: [**http://160592857366.free.fr/joe/ebooks/ShareData/Heuristics%20for%20the%20Traveling%20Salesman%20Problem%20By%20Christian%20Nillson.pdf**](http://160592857366.free.fr/joe/ebooks/ShareData/Heuristics%20for%20the%20Traveling%20Salesman%20Problem%20By%20Christian%20Nillson.pdf)

The above source is the one that guided my findings the most. It explained each algorithm step by step, and provided time complexity and efficiency in terms of percentage worse than optimal.