Abstract

Contents

1 Introduction

The Travelling Salesperson Problem (TSP) is an NP-Hard problem which is commonly found in combinatorial optimisation, theoretical computer science and operations research. It's a simple problem on the surface which simply states that "Given a list of cities and distances between each pair of them, what is the shortest possile route that visits each city exactly once and returns to the origin city?" [1]

This question appears deceptively simple, however with large sets of cities the search space becomes incredibly large, and as such computation time can be astronomical, since before you even start trying to compute the shortest tour, there are n! different combinations to compute, assuming a complete graph. Clearly this meant another angle had to be taken for this problem.

As such many people decided to go for a more heuristic approach towards solving the TSP problem. So instead of trying to find the optimal solution with an infeasibly long time span, we would settle for a 'good enough' solution with a much lower time complexity, however since TSP is a combinatorial optimisation problem it is much more difficult than continuous optimisation, with higher chances of falling into local optimum rather than the global optimum.

And yet even with this said, there exists an algorithm that has been created more than 30 years ago (1976) that has been proven to be at most 50% worse than the optimal solution for any given TSP problem and is known as Christofides algorithm. This algorithm has remained the best approximation available for over 30 years, and only recently in this year (2020) has another, slightly more efficient algorithm been found [3], although it has yet to be verified even if the general consensus is that it seems correct.

However another way of finding approximate solutions is becoming more and more popular over time, and has its roots deeply set in evolution and natural selection, hence the name they are given: Evolutionary algorithms. In essence these algorithms replicate the generational aspect of natural selection, taking the best individuals from a given population and 'breeding' them, these 'children' can then be tested to see if they perform any better than their parents, and if so replace the worst individuals in the population, and this process continues to repeat until a given goal is complete, whether that turns out to be an actual solution being found, until a good enough solution is found or after a certain number of generations have been looped through.

Genetic algorithms have already been used to try and solve the TSP which shall be discussed later, yet the major point of genetic algorithms is that you start with an entirely random population with which to create new solutions with a large degree of randomness in terms of the search operators. If however we were to replace one of the population with a strong approximate solution (x), how would that affect the algorithm? Theoretically in the worst case scenario this genetic algorithm would simply return x, since if no better one can be found, then x would remain in the population as the best solution, so we are no worse off compared to when we started the algorithm. However in the best case scenario the algorithm would produce a better approximate solution than x and return this instead.

As such, in this thesis I will be investigating the usage of both the mathematical and algorithmic methods of finding a solution to the Travelling Salesman Problem, and thereby seeing if these methods can be combined into a singular algorithm that can find a more optimal solution.

2 Background

2.1 Christofides Algorithm

Even with the algorithm having been created over 30 years ago in 1976, there has yet to be another verified algorithm at this time that can produce more optimal approximate solutions to the TSP problem than this. This algorithm has been proven to be at most 50% worse than the optimal solution whilst having an $O(n^3)$ time complexity where n is the number of cities for which we are trying to solve the problem. [4] If we instead look at the best algorithm currently available to solve the TSP (aka the Held-Karp Algorithm) we can see that its complexity is $O(n^2 2^n)$ [2] which, whilst astronomically better than the original $O(n!)$, is significantly worse than the approximate solution that Christofides algorithm provides.

This is often the case with such computationally expensive problems, a balance must often be made between finding the optimal solution, and finding a solution in an appropriate amount of time. The Held-Karp algorithm focuses more on finding the optimum answer rather than finding it in a feasible amount of time, whereas Christofides Algorithm was designed to find as best an approximation as it can whilst still having a relatively feasible computation time which is what can make it so appealing to use.

One of the strong points about Christofides algorithm is just how simple it is, having only 4 or 5 major steps (many sources combine steps 4 and 5 whilst others do not [5, 6]) even if these steps are more likes processes themselves.

Nonetheless the steps to this algorithm are as follows:

1. Find a minimum spanning tree T.

2. Find a minimum weight perfect matching M for the odd degree vertices in T.

3. Calculate $M \bigcup T$.

4. Find an Euler tour of $M \bigcup T$.
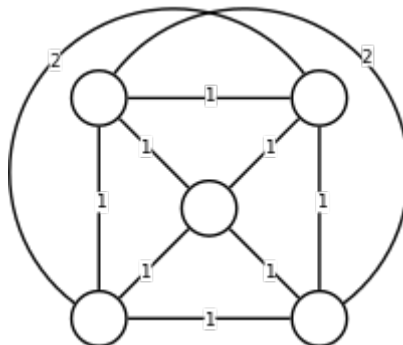
5. Remove any repeated vertices.

Figure 1: Full Graph

As stated the process is compressed into few steps and is quite simple to understand, and yet creating this algorithm within a programming language is more complex than one would initially assume. Finding a minimum spanning tree (Figure 2) of the full graph (Figure 1) is quite simple, the two most common algorithms known for solving this are Prims and Kruskals, both of which are effective methods with time complexities of $O(n)$ and $O(n^2)$ respectively [7] though choosing one or the other is mostly affected by the number of edges within the graph we are finding the minimum spanning tree for, with Prims being more suitable if there are a large number of edges, and Kruskals if not.
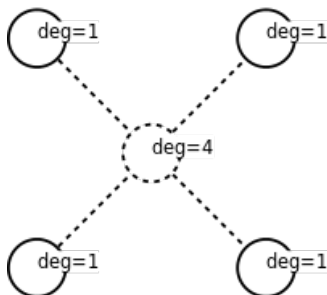

Figure 2: Minimum Spanning Tree

The minimum weight perfect matching of T (Figure 3) is accomplished by finding $\frac{T}{2}$ edges which 'link' together every vertex in T by the shortest edge possible. For this one of the most common algorithm in use appears to be the blossom algorithm and has a time complexity of $O(n^2 m)$ in the worst case where n is the number of vertices and m is the number of edges [8].
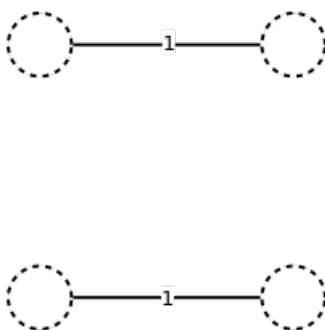

Figure 3: Minimum Weight Perfect Matching

From here we simply calculate the union of M and T (Figure 4), including any of the repeated edges that may be present, before moving onto the Euler tour. A Euler tour is simply a tour of the graph which visits each edge exactly once, regardless of starting or ending vertex (Figure 5).
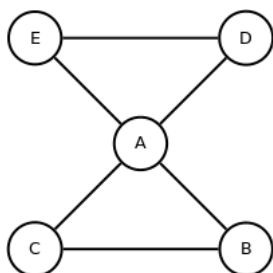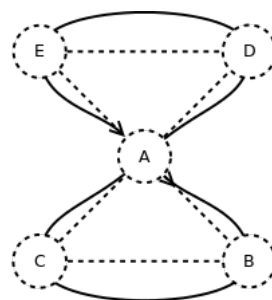

Figure 4: $M \bigcup T$

Figure 5: Euler Tour

Then need to remove any repeated vertices within our tour and replace them with the direct edge between those vertices, and once this is done we have the algorithms output (Figure 6).
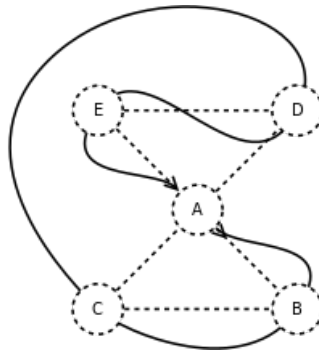

Figure 6: Christofides Algorithm Output

(Figures 1-6 taken from: Wikipedia: Travelling Saleman Problem [1])

With this algorithm output we now have a solution to the TSP for our specific graph, and this solution is guaranteed to be within 50% of the optimum [4], which is significantly better than trying to calculate the true optimum for a very large graph in an infeasible amount of time. However there is clear room for improvement, hence why other methods have been designed and tested.

2.2   Genetic Algorithms

Genetic algorithms are a subsection of Evolutionary algorithms

References

[1] Wikipedia: Travelling Salesman Problem
    https://en.wikipedia.org/wiki/Travelling_salesman_problem (Accessed 20 October 2020)

[2] Hutchinson, C. et al. (no date) CMU Traveling Salesman Problem, p. 25. Available at: https://www.math.cmu.edu/~af1p/Teaching/OR2/Projects/P58/OR2_Paper. pdf (Accessed: 27 October 2020)

[3] Klarreich, E. (no date) Computer Scientists Break Traveling Salesperson Record, Quanta Magazine. Available at: https://www.quantamagazine.org/computer-scientists- break-traveling-salesperson-record-20201008/ (Accessed: 22 October 2020).

[4] Christofides, N. (1976) Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem. CARNEGIE-MELLON UNIV PITTSBURGH PA MANAGEMENT SCIENCES RESEARCH GROUP. Available at: https://apps.dtic.mil/sti/citations/ADA025602 (Accessed: 22 October 2020).

[5] Sitters, R. (no date) Chapter 2: Greedy Algorithms and Local Search. Available at: https://personal.vu.nl/r.a.sitters/AdvancedAlgorithms/2016/SlidesChapter2- 2016.pdf (Accessed: 27 October 2020).

[6] Christofides algorithm (no date). Available at: `https://xlinux.nist.gov/dads/HTML/christofides.html` (Accessed: 27 October 2020).

[7] Huang, F., Gao, P. and Wang, Y. (2009) Comparison of Prim and Kruskal on Shanghai and Shenzhen 300 Index Hierarchical Structure Tree, in 2009 International Conference on Web Information Systems and Mining. 2009 International Conference on Web Information Systems and Mining, pp. 237241. doi: 10.1109/WISM.2009.56.

[8] Kolmogorov, V. (2009) Blossom V: a new implementation of a minimum cost perfect matching algorithm, Mathematical Programming Computation, 1(1), pp. 4367. doi: 10.1007/s12532-009-0002-8. Available at: `http://mpc.zib.de/archive/2009/1Kolmogorov2009_Article_BlossomVANewImplementationOfAM.pdf` (Accessed: 29 October 2020)

Appendices