Final Project

Ronney Aovida

8/5/2016

ISC4221

Solving The Traveling Sales Man Problem

Purpose:

The purpose in this research is the testing of a different approach to a currently not optimally solvable problem in the world of computational sciences. The Traveling Sales Man problem is a classic algorithmic problem focused on optimization, in other words, finding a solution that is the cheapest. It is typically expressed as a graph describing the locations of a set of nodes and end goal is to simply arrive at every single node on said graph ONCE, and strictly only once, until it comes back around and connects back to the original starting node, creating a route. This alone is a relatively simple problem to solve, however there is always a catch, the distance between each node on the graph contains a certain weight. The goal of the problem is to design a route that can connect each node only once but in an order that returns the least amount of weight in relation to all other possible routes. Another way to view this problem is described as a salesman who must travel between N cities, his goal is to visit each city only once since he is on a tight schedule and must go to a different city every day. The salesman however is very tight on money so his way of transport will have to be very efficient while traveling, whether it be by train, plain or automobile. The cost to go to each city is different depending of type of travel and distance, by taking these variables and placing a weighted value on each the sales man is now faced with the problem of figuring out which route would return the least weighted cost between traveling cities.

Scope:

The scope of this work is in regards to many real life computational works:

* such as in the manufacture of a circuit board. It is important to determine the best order in which a laser will drill thousands of holes, An efficient solution to this problem reduces production costs for the manufacturer.
* Used as a test case for almost every new discrete optimization algorithm such as branch and bound.
* For integer and mixed-integer algorithms.
* Local search algorithms.
* Simulated annealing, genetic algorithms and DNA computing

Approaches to solving this problem:

This problem can be solved in numerous ways using different algorithms, to name a few:

* Construction heuristics: nearest-neighbor, MST, Clarke-wright, Christofides
* Simulated annealing, Tabu search, the Held-Karp lower bound
* Lin-Kerninghan-Helsgaun and exact methods using integer programming.

A different approach:

The approach I decided to take in solving this problem does not involve any of the above mentioned algorithms, but rather the use of an algorithm being similar to them. The essence of this problem is to find the shortest possible path through the nodes in relation to the weighted edges. I found that a quick and efficient way to find a shortest path through a set of nodes is through the use of a minimum spanning tree. A minimum spanning tree is a spanning tree of a connected, undirected graph. It connects all the vertices together with the minimal total weighting for its edges. This spanning tree will provide the most optimal route of the problem in relation to the edges weights. However, it does not create a complete path to every node that routes back to the original starting point. In order to solve this dilemma we simply have to calculate the distance between the starting node and the closest node to it that is not a part of the spanning tree output already and add that edge to the solution.

Attaining the minimum spanning tree:

In order to calculate the minimum spanning tree of the graphed problem we need to use a algorithm that can calucate the shortest path between nodes without using the same node twice. The algorithm chosen to complete this task is called Kruskals Algorithm.

How Kruskals Algorithm works:

The algorithm will first begin from a chosen starting point. the edges choosen are the shortest unused edge of the edge list that does not create a cycle (a cycle is a term to refer to the use of two Nodes from a given edge more than once in a spanning tree). as long as the two Nodes from a given edge do not occur more than once in the tree then a cycle is avoided. This will continue to work through every node in the graph until every node has been touched atleast once, resulting in a non-complete route around the graph outlining the skeleton of the shortest possible path.

Putting it all together:

To solve the TSP using this approach will require the user to enter an input of vectors (nodes) and coordinates of those vectors. These inputs are then formatted into a edge list of every possible edge between all vectors, such as: (A,B),(A,C),(A,D),(B,C),(B,D) and so on. It is then used as input into Kruskals Algorithm, where every node will be checked for the lowest weighted value on their edges and making sure no cycles are being formed. The output will look something like (4,A,B),(5,B,C),(2,C,E) and so on.. forming a minimum spanning tree. This minimum spanning tree is then checked to see which node within it is closest to the starting node and makes sure that the edge between them has been unused, if all the criteria is met, that final edge is added to the minimum spanning tree output to form a solution to the Traveling Sales Man problem.

Comparing Different Approaches to this problem:

There are many different approaches to this problem as stated above, however each approach has its downfalls and benefits.

* A algorithm that is very commonly used to solve this problem due to its ease of use is the nearest-neighbor heuristic (a greedy algorithm). This method simply goes to the nearest edge with the lowest weights from the starting node and from the node connected to that edge it does the same thing until it reaches the end node. The benefit of using this approach is the extremely fast run time, however the problem with it is that it is typically the most inaccurate approach to solving the problem since it will take an extremely long route to a node just so that it can continue to take a bunch of small edges that add up to be more than a single edge that is relatively large but leads the route directly to the end.
* The most widely Used algorithm to solve the TSP is known as the branch and bound algorithm. This algorithm is one of the more complicated algorithms out there for solving this problem. The down side to using this algorithm is that is has a very slow run time compared to other methods, however it does make up for that by providing the most accurate optimal solution among almost all other methodologies.
* The approach of using a minimal spanning tree is in the goldilocks zone in regards to its accuracy and run time. it may be more slow than the nearest neighbor heuristic but faster than the branch and bound, its accuracy however is much more optimal than the nearest-neighbor but of course not at the same level of accuracy as the branch and bound algorithm provides.

Conclusion:

The TSP to this day remains to be one of the few problems that have yet to be fully solved since any algorithm today has yet to reach a 100% accuracy in efficiency compared to solving it by hand.

One day in the future it is highly predicted that an algorithm will reach the optimal accuracy our human brains are capable of or perhaps even better. Until then using any of the previously stated algorithms should suffice for saving a lot of time since our human brains are much slower than any algorithm, at the expense of being more accurate that any algorithm as well.

**References/sources**

* [**https://www.seas.gwu.edu/~simhaweb/champalg/tsp/tsp.html**](https://www.seas.gwu.edu/~simhaweb/champalg/tsp/tsp.html)
* [**https://simple.wikipedia.org/wiki/Travelling\_salesman\_problem**](https://simple.wikipedia.org/wiki/Travelling_salesman_problem)

**// This is a Draft submission. A Submission with images and a little more detail will follow.**