**Problem 5**

We used a Branch and Bound algorithm to address the Traveling Salesman Problem. We select a city at random, and then recursively "branch" to include and exclude all possible paths. This exhaustive search is greatly reduced by the "bound" portion which involves keeping track of the best path discovered thus far and ignoring all subsequent sets of paths that have no chance of beating the current best.

The following optimizations were applied. These can be identified easily in the code as they are numbered the same as below.

<u>Optimization 1</u>: The "bounding" of Branch and Bound as described above.

<u>Optimization 2</u>: Every vertex contains a collection of all edges connected to it. Those edges are pre-sorted in order of shortest to longest length. This enables the algorithm to produce a short path quickly at the beginning of its processing thereby allowing it to bound, or truncate, more of the remaining execution tree. Empirically, this was confirmed by excluding this sort and also by reversing it. Ascending order appeared to be best.

<u>Optimization 3</u>: While building the execution tree, we choose to examine only the shortest few edges (not counting the one that has already been selected) that connect to each vertex, as specified by the 'breadth limit'. The breadth limit is a hyper-parameter specified manually at the start of the execution. This greatly reduces the overall execution tree.

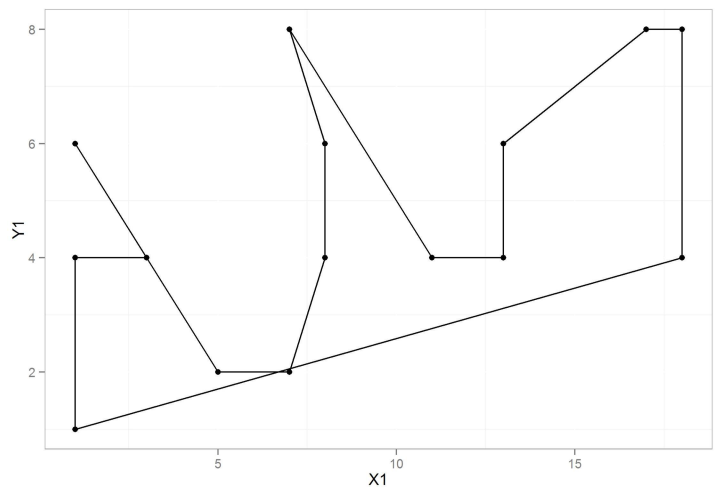
There is a risk to this optimization in that it is possible that no solution could be found. In fact, with a low enough breadth limit, it is almost guaranteed that no solution could be found. For this reason, the breadth limit is not a hard cutoff. If no viable solution has been found among the children of the vertex being considered, then the algorithm explores longer paths until one viable solution is found.

**The Code**

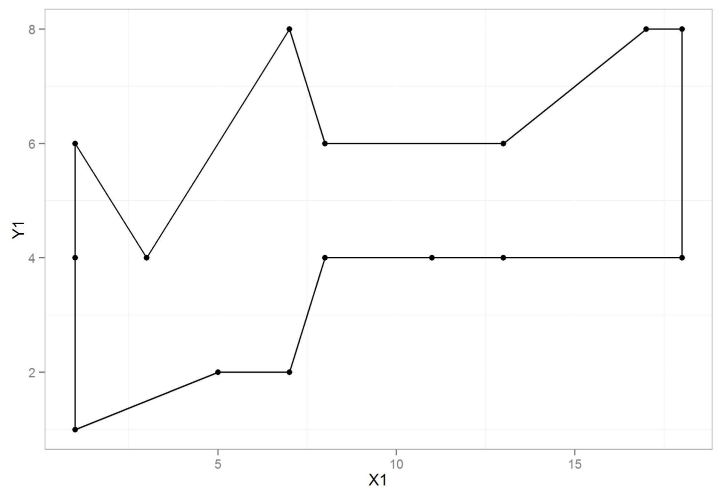
The code is attached. It was written in C# and contains three classes: TspSolver which oversees the process, Vertex, and Edge. Some aspects of the code were not industrial quality and this is mentioned in the comments.

**The Results**

On a synthetic test set consisting of 15 cities with no breadth limit set, a solution 13.6% worse than the best possible was found within 0.01 seconds:



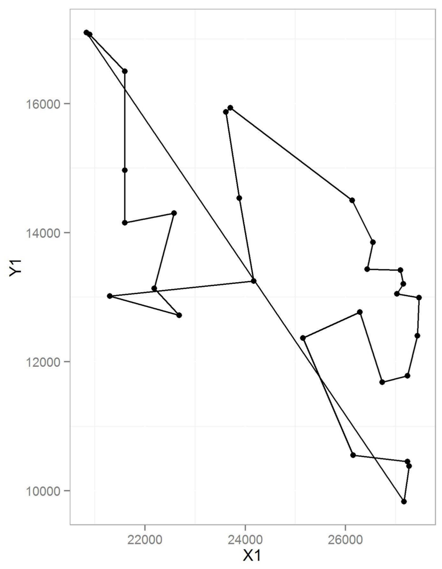
Within 6.45 seconds, it found the exact solution:



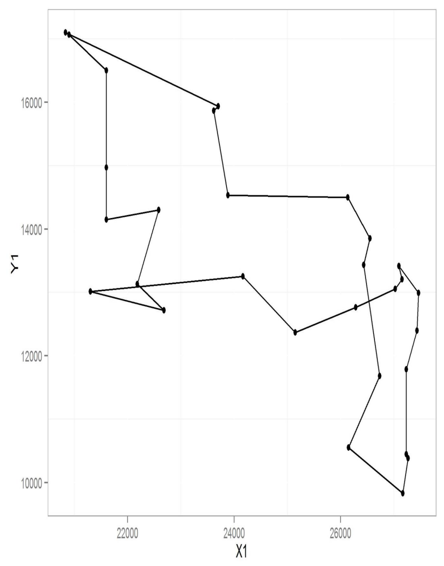
However, it took 78 seconds of exhaustive search to confirm that this was indeed the exact solution.

With a breadth limit of 6, this same solution was found in 6 seconds and confirmed within 15 as the best the algorithm could produce.  However, when the breadth limit is set, there is no exhaustive search, and therefore no final confirmation that our best solution is the exact solution.

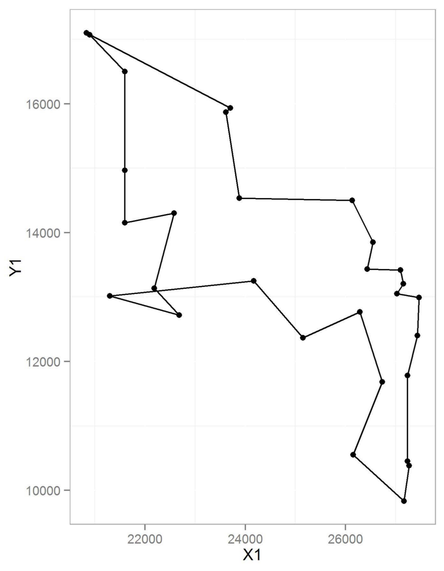
Using data found at http://www.math.uwaterloo.ca/tsp/world/countries.html, we ran our algorithm on the 'Western Sahara' data set with 29 cities using a bread limit of 6. Within 0.01 seconds we found a solution 32% worse than optimal:



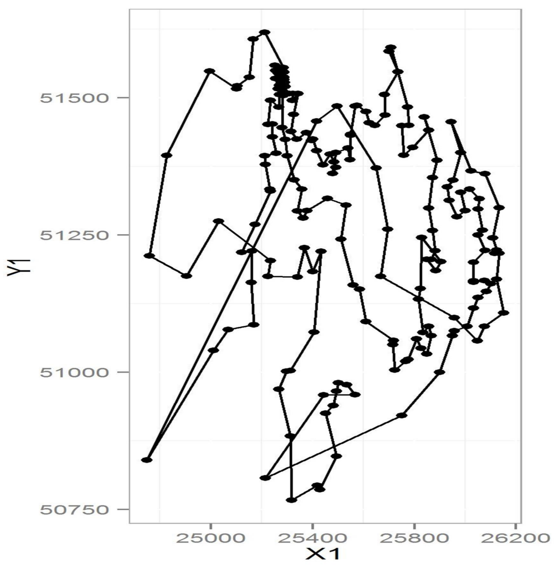
Within 38 seconds, 10% worse than optimal:



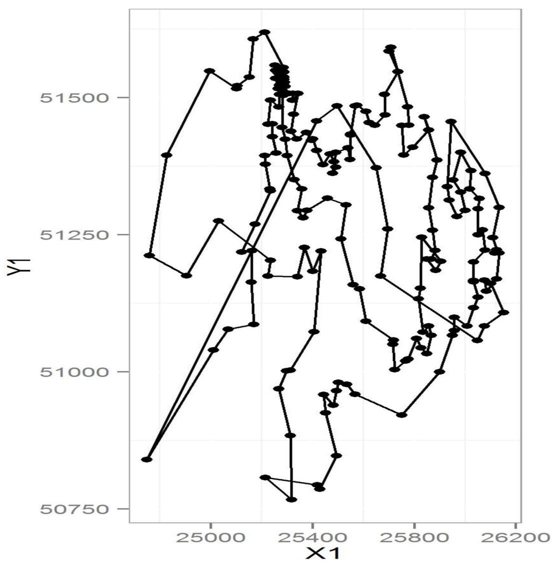
And after 313 seconds, 6.7% worse than optimal:



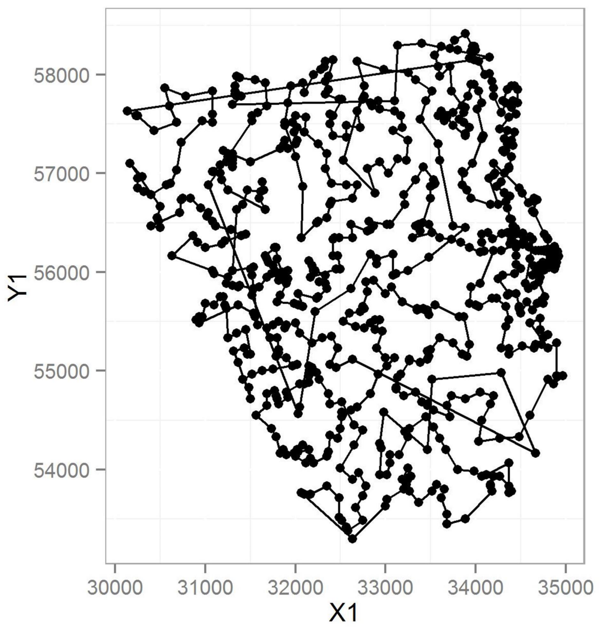
We can experiment with the breadth limit. A lower number gives us a less good final solution, but gets us to a pretty good solution much quicker than a higher number does. Using the Qatar data set with 194 cities, a breadth limit of 3 gave us a solution 22.9% worse than optimal in 50 seconds.



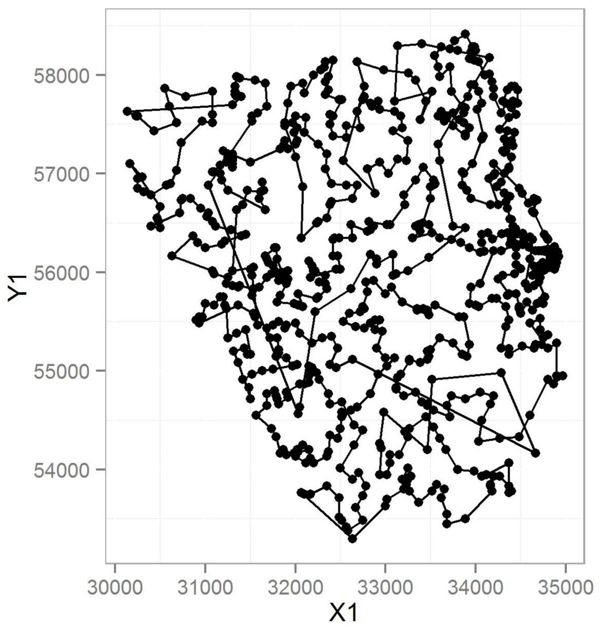
Whereas the more restrictive breadth limit of 2 gave us a better solution, 19% worse than optimal, in only 7.5 seconds:



Using the Uruguay data set with 734 cities, using a breadth limit of 4, we obtained a solution 27.6% worse than optimal in 0.01 seconds.

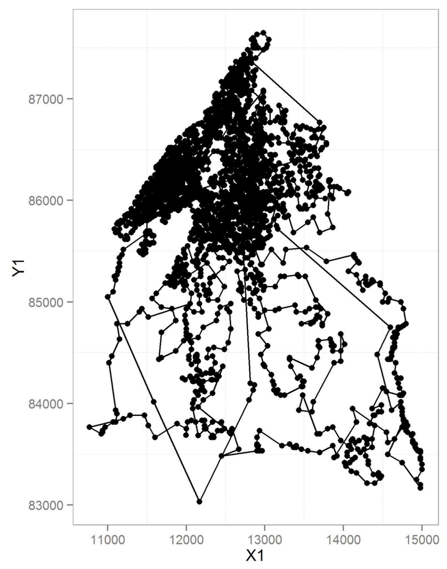


In 10.97 seconds, we improved this to a solution 23.4% worse than optimal:



Another two hours of execution did not yield any better results than this.

Finally, using the Oman data set with 1979 cities and a breadth limit of 4, we obtained a solution 39.1% worse than optimal in 4.28 seconds:



**Compiled Results**

| Test Set | Number of Cities | Breadth Limit | Duration (secs) | Path Found | Actual Best | Overage |
|---|---|---|---|---|---|---|
| Synthetic | 15 | unlimited | 0.01 | 55.2 | 48.6 | 13.6 |
| Synthetic | 15 | unlimited | 6.45 | 48.6 | 48.6 | 0.0 |
| Synthetic | 15 | 6 | 0.01 | 58.3 | 48.6 | 20.0 |
| Synthetic | 15 | 6 | 6.32 | 48.6 | 48.6 | 0.0 |
| W. Sahara | 29 | 6 | 0.01 | 36388 | 27603 | 31.8 |
| W. Sahara | 29 | 6 | 48 | 30313 | 27603 | 9.8 |
| W. Sahara | 29 | 6 | 313 | 29458 | 27603 | 6.7 |
| Qatar | 194 | 3 | 50 | 11497.28 | 9352 | 22.9 |
| Qatar | 194 | 2 | 7.49 | 11128.44 | 9352 | 19.0 |
| Uruguay | 734 | 4 | 0.01 | 100919.8 | 79114 | 27.6 |
| Uruguay | 734 | 4 | 0.36 | 97607 | 79114 | 23.4 |
| Oman | 1979 | 4 | 4.28 | 120895 | 86891 | 39.1 |

With optimization, the Branch and Bound technique, along with various optimizations including our Breadth Limit, is viable for producing 'pretty good' results.  Although we could have obtained faster results with multi-threaded code, we would not be likely to see an order of magnitude improvement. Therefore, this approach is not viable for producing an exact solution in reasonable time.