CS 325-401
Summary of TSP Results

Compile on flip with: make proj4

Test Case	Minimum Tour length	Time
1	7833	0.000000s
2	8966	0.000000s
3	13468	0.000000s
4	22538	0.030000s
5	33275	0.180000s
6	46411	0.720000s
7	74684	4.800000s

Susan Kuretski
Ava Petley
Robert Sparks
CS325-401, Fall 2015
Project Group 2

Project 4 Report
Traveling Salesman Problem

Introduction:

"Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?" (Wikipedia, 2015).

The traveling salesman problem, TSP, is famously known for its difficulty, specifically listed as NP-Hard. A brute force algorithm takes O(n!) time to solve the TSP, making it superbly beyond polynomial time. In addition, with current computing power, to solve brute force even with a relatively small n will most likely be impractical and/or infeasible.

This project is an implementation of heuristics to find a potential solution to the TSP. We intend to use Christofides algorithm (a construction algorithm) along with 2-OPT (an improvement algorithm) to optimize a potential solution.

Christofides algorithm is a "$O(n^3)$ heuristic algorithm [used] for solving n-city travelling salesman problems (TSP) whose cost matrix satisfies the triangularity condition" (Christofides 1976). The ratio of the worst-case analysis to the optimal solution is strictly less than 3/2. Contrarily, a lower bound of TSP can be described by the Held-Karp algorithm, where it is always at least ⅔ of the optimum (Williamson 1990).

2-OPT can be used as an improvement algorithm using simple local searching where two edges are removed from the tour and then reconnected to create two new edges if the new tour will be shorter than the original. 3-OPT is essentially the same process, but with three edges. And similarly, k-opt can be utilized, but generally only provides solutions that are slightly better than 2- and 3-OPT. 2-OPT will generally result in a tour 5% above the Held-Karp lower bound (Nilsson 2003).

By combining Christofides with an upper bound of 3/2 and 2-OPT improvement, the aim is to provide a solution bounded above by 1.25.

Description of Algorithm:

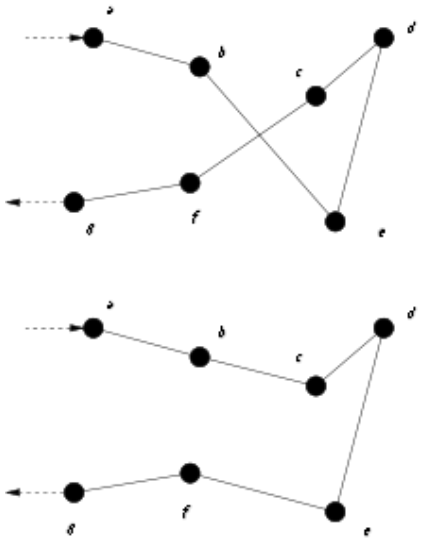
Influence from: Approximation algorithms 2001.
Given a nonnegative edge weight graph G = V,E, is there a Hamiltonian cycle of minimum cost? We used Christofides algorithm combined with 2-OPT.

Christofides

1. Find T = the Minimum Spanning Tree of graph G
2. Find O = vertices with Odd Degree from T
3. Find P = the minimum cost perfect matching from O
4. Find G' = T U P
5. Find E = the Euler tour of G'
6. Find the Hamiltonian cycle of E

2-OPT

1. Select edge (v_1, v_2) and (v_3, v_4).
2. Replace edges with best pair such that it minimizes tour length.



[1] 2-opt visualization

**In the above picture, edge(b,e) and edge(c,f) are swapped to create edge(b,c) and edge(e,f).

Pseudo-code:

For Minimum Spanning Tree

References to CLRS pg 634

```
MST-Prim(G, w, r)
      for each u ∈ G.V
            u.key = infinity
            u.pi = NIL
      r.key = 0
      Q = G.V
      while(Q ≠ empty set)
            u = Extract-min(Q)
            for each v ∈ G.adj[u]
                  if v ∈ Q and w(u, v) < v.key
```

[1] Image courtesy of https://en.wikipedia.org/wiki/2-opt

```
                v.pi = u
                v.key = w(u, v)
```

Find Vertices with Odd Degree

```
Find-Odd(G)
     for each v ∈ G
          if(sum of edges∈G.adj[u] is odd)
               v = odd degree
```

Minimum Weight Matching

```
Min-Matching(O)
     while(O ≠ empty set)
          v.prev = infinity
          for each v ∈ O
               if(v.next < v.prev)
                    v.prev = v.next
          M.push(edge)
          O.pop(v)
     return M
```

Union of MST and MWM

```
Union(T, M)
     for each e in T
          U.push(e)
     for each e in M
          U.push(e)
     return U
```

Euler Tour

```
Find-Euler(AdjMatrix)
     while(stack ≠ empty set || size of adjMatrix[i] > 0
          if(size of adjMatrix == 0)
               path = i
               next = stack.top
               stack.pop
               i = next
          else
               path = i
               neighbor = last element of adjMatrix[i]
               remove edges of (u,v) and (v,u) in adjMatrix
               stack.push(i)
               i = neighbor
     return E
```

Hamiltonian Cycle

```
Ham-cycle(G, path)
      H.push_back(path[0])
      for i = 0 to path.size
            for k = 0 to H.size
                  if path[i] == H[k]
                        delete path[i]
                  else
                        H.push_back(tour[i])
      return H
```

2-OPT

```
2-OPT(Path)
for j = 0 to path.length
      for i = 0 to 4
            temp[i] = path[j]
      current = distance(temp[i], 4)
      swap values in temp array
      newPath = distance(temp[i], 4)
      if(newPath < current)
            newPath = current     //new path replaces old part of path
      j++                         //continue on another path section
```

Distance of a path (n is used to specify either whole path or portions of a path)

```
Distance(Path, n)
      for i = 0 to n
            u = path[i]
            if(i < n - 1)
                  j = i + 1
                  v = path[j]
            else
                  v = path[0]
            distance = e.path[u][v]
            total += distance
      return total
```

Best Tours for 3 Example Instances:

Example 1

Distance	Time	Ratio
126000	0.00000s	1.19

Example 2

Distance	Time	Ratio
3486	0.010000s	1.37

Example 3

Distance	Time	Ratio
2223398	55.700001s	1.45

Best Tours for Competition Cases:

See page 1.

References

Approximation algorithms. c2001. Princeton, NJ: Princeton University; [accessed 2015 Dec 01]. http://www.cs.princeton.edu/~wayne/cs423/lectures/approx-alg-4up.pdf

Christofides, Nicos. Worst-case analysis of a new heuristic for the travelling salesman problem. Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., et al. (2009)[1990]. Introduction to Algorithms, Third ed. MIT Press and McGraw-Hill.

Nilsson, Christian. Heuristics for the traveling salesman problem. *Tech Report*, Linkoping University, Sweden, 2003.

Travelling salesman problem. c2015. Wikipedia; [accessed 2015 Nov 27]. https://en.wikipedia.org/wiki/Travelling_salesman_problem

Williamson, David Paul. Analysis of the Held-Karp heuristic for the traveling salesman problem. Massachusetts Institute of Technology, Cambridge, MA, 1990.