The Traveling Salesman Problem

Problem:

In an effort to conserve resources, a salesman wishes to find the shortest route that connects all points of interest and returns him home again.

Why Is This Problem Important?

- In the realm of Computer Science, this problem is classified as NP-Hard. NP-Hard Problems share characteristics such that there exists a non-deterministic algorithm that operates in polynomial time.
- Also, all NP-Hard Problems are capable of being reduced into one another, therefore a solution to one will solve every one, such as an encryption breaking algorithm. To date, there exists no known solution to any NP-Hard problems.
- Many businesses with travel in mind are actively searching for a solution: Airline Industry, Couriers, etc.
- Dates back to 1800's by an mathematicians Sir William Rowan Hamilton and Thomas Kirkman. Additionally, it was studied by Karl Menger in 1928.
- In March 2005, 33,810 points on a circuit board were traced with a tour length 66,048,940 units taking over 15.7 CPU years (meaning if two CPUs threaded this task, it would take 7.83yrs to complete)!
- Exact solution require O(n!) time, making large tours infeasible to calculate. Therefore, approximations are often used.

Nearest Neighbor:

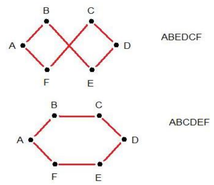
A "very naïve" algorithm that most programmers employ. It is very simple to implement.

1. Designate Start Location and add to tour
2. While not all locations in tour, add closest next location to last in tour
3. Add start location again to return.

Complexity: $O(n^2)$

Two-Opt:

It is easy to tell, however, if two paths cross, it's not optimal, therefore "un-cross" them.



1. Start with an initial Path p
2. For each city, find best exchange (see picture) that improves path and change path

Complexity: $O(n^2)$

Simulated Annealing:

In nature, a heated system becomes more stable as it cools. This algorithm is analogous to this phenomenon. In SA, the probability of making a bad swap is dependent on how "hot" the system is; this prevents reaching a local minima that prevents optimal updates.

1. Start with initial path p_0, temperature $t=t_0$, cool down rate $r < 1$, best path b
2. While $t > t_{min}$
   A. Select two random cities
   B. If swapping improves (same as 2-opt exchange), swap. That is, if length(p) < length(b), b = p
   C. Else, swap given probability $e^{-d/t}$ where d = distance added to path
   D. Update t = t * r
   E. Result is b

Complexity: $O(n*\log_r[t_{min}/t_0])$