Nicholas Milford
March 7

CS 420 Final Project: Travelling Salesman

2-opt

The 2-opt algorithm is a local search iterative improvement travelling salesman approximation. Given an existing tour, it repeatedly rearranges pairs of edges, modifying the tour as such if the change leads to a tour of lesser length. This algorithm can be further generalized into a k-opt algorithm, where k edges are removed and rearranged, and the best rearrangement is compared against the existing tour. While intermediate values of k can lead to fewer iterations to reach the local optimum (i.e. 3-opt), this wasn't necessary to complete the largest problem in the five minutes given.

Because 2-opt is a local search, concepts explored in artificial intelligence can be applied for a more optimal solution at the expense of more time. This idea also wasn't necessary, because the algorithm was able to meet the 1.25 * optimal bound without further improvements.

Because 2-opt is an iterative improvement algorithm, many iterations are able to be saved by starting with an approximate solution. The nearest neighbor algorithm creates the initial tour, and then 2-opt improves the nearest neighbor solution to the local optimum.

Pseudocode:

```
1. Create an initial tour T
2. While the tour has been improved in the last iteration:
    a. For every pair of edges $e_1 = (v_{11}, v_{12})$, $e_2 = (v_{21}, v_{22})$: ( $O(n^2)$ )
        i. Create a new tour T' where $e_1$, $e_2$ are replaced by $e'_1$, $e'_2$, where $e'_1 = (v_{11}. v_{21})$ and $e'_2 = (v_{12}, v_{22})$
        ii. If cost(T') < cost(T):
            1. T = T'
3. Return T
```

Total running time: $O(n^2$ * iterations to reach local minimum)

Nearest Addition

The nearest addition algorithm is a greedy algorithm that operates on the same principle as Prim's algorithm. The vertex to be added is selected from the set of vertices not already in the tour based on distance to any vertex that is part of the tour. The closest vertex is added, and the tour ordering is rearranged to match. This gives more optimal results than the Nearest Neighbor or Nearest Fragment algorithms, but runs in $O(n^3)$ time, rather than $O(n^2)$.

Pseudocode:

```
1. Start with a tour T consisting of one node
2. While the tour is not complete: ( $O(n)$ )
    a. Find the vertex nearest to any vertex already in the tour ( $O(n^2)$ )
    b. Add that vertex to T before or after the vertex in the tour
3. Return T
```

Total running time: $O(n^3)$

Christofides

Christofides algorithm is suggested in class as an algorithm that should be investigated. It is the most complex to implement, but guarantees a solution within 1.5 times optimum in a known amount of time. Christofides algorithm constructs the salesman tour from a minimum spanning tree and some other heuristics.

Implementation

Nearest Addition and 2-opt were selected to be implemented mostly because of time constraints. They each are built on very simple steps and thus are easy to implement, whereas Christofides algorithm requires several steps which each have complex implementations themselves (i.e. minimum spanning tree, eulerian tour).

Results

2-opt, example 1: 117634 (1.09 * optimal), 0.152 seconds
2-opt, example 2: 2856 (1.07 * optimal), 0.222 seconds
2-opt, example 3: 1728126 (1.11 * optimal), 58.8 seconds

Nearest Addition, example 1: 132040 (1.22 * optimal), 0.143 seconds
Nearest Addition, example 2: 3499 (1.35 * optimal), 0.216 seconds
Nearest Addition, example 3: 2023774 (1.29 * optimal), 1 hour, 1 minutes, 44 seconds

2-opt had the best results in the least amount of time in all examples.

Test case 1: 5676, 0.127 seconds
Test case 2: 8927, 0.142 seconds
Test case 3: 13403, 0.217 seconds
Test case 4: 19304, 0.321 seconds
Test case 5: 26886, 1.12 seconds
Test case 6: 35715, 0.693 seconds
Test case 7: 55608, 3.778 seconds

2-opt also had the best results in the least amount of time for all test cases

Resources

http://pedrohfsd.com/2017/08/11/2opt-part2.html
https://personal.vu.nl/r.a.sitters/AdvancedAlgorithms/2016/SlidesChapter2-2016.pdf
https://en.wikipedia.org/wiki/Christofides_algorithm
https://en.wikipedia.org/wiki/Travelling_salesman_problem