David Demoll

CS320P06 Complexity Analysis

The sub problem is g(i, S) is defined as the shortest path from vertex i, visiting all the vertices in set S, and ending at vertex 1. When S is empty, g(i, S) = M[1][i].

M[i][j] is defined as the distance between nodes i and j.

Pseudocode for algorithm:

For k = 1 to |S|

For all subsets of S of size k

For each vertex j in subset of S

g(i, S) = min{g(j, S – {j} ) + M[i][j] j∈S, i≠j}

return min[g(i, S) + M[i][1]]

First for loop runs in approximately n time.

Second for loop runs in

= Θ(2n)

1/k! = Θ(1) because k is constant

Third loop:

Each subproblem g(i, S) is computing s-1 numbers each in linear time, which equals O(n) per subproblem.

Putting all three loops together the average case time complexity is Θ(n22n).

My dynamic programming implementation uses a recurrence relation in terms of the subproblem g. There are at most O(n2n) subproblems and each one takes linear time to solve, therefore the total running time is O(n22n ). Since the time complexity is exponential, this algorithm is infeasible for even a slightly high number of vertices. In my experience, any number of vertices greater than 20 did not work.

I start by looping through all subsets *s* of size S-1 and calculating g(i, *s*) + M[i][j]. g(i, *s*) calls itself recursively until the base case where *s* is the empty set is reached. In the base case, the value of g(i, *s*) is simply M[i][1], the cost from the first vertex to i. Then, for each call to g we take the minimum cost path and use that value to build back up to the top of the call stack, which returns the optimal Hamiltonian cycle. My algorithm recovers the optimal Hamiltonian cycle by remembering the values of j used to achieve the minima of function g for set |S| = 1..n using function p.

In the case where a call to g with the same arguments has already been calculated, memiozation is achieved using the unordered\_map ADT, which is implemented using a hash table. This allows the results of the function calls to be saved, optimizing the solution by looking up saved function calls in O(1) time.