Alden Liu
Kelvin Zhang
Darren Wang

CS170 Design Doc

In the conquering kingdom problem we begin at the starting node and at this/each node consider both the travel costs and whether to conquer it. To tackle both tasks we reduce the given conquering problem from a traveling salesman variant, the traveling purchaser problem. In the traveling purchaser problem we are given a list of items we wish to obtain(G), a list/set of possible supermarkets (S_n), a matrix C where (C_{ij}) is the traveling cost going from node i to node j, and a I_n which is a list of items found at supermarket S_n. In the traveling purchaser problem, we are trying to find the optimal route to buy all the items you want in the least amount of money considering both travel cost and purchasing price. Each market has its own respective items with their own respective costs. So its dynamic programming approach is to find the optimal markets to go to and buy certain items.

In our dynamic programming approach,we represent G as all the kingdoms to replicate conquering every kingdom. Each of the kingdoms simulates a supermarket S_n where the items I_n available at that supermarket is itself and the other neighboring kingdoms returned by the N(k) with the total cost of all the items only being the conquering cost thus replicating the effect of the surrounding kingdoms surrendering.

In the conquering problem we come across the problem on needing to buy a node that we already have gained from purchasing the neighboring node; however, in our algorithm we plan on allowing our algorithm to buy multiple amounts of the same kingdom as long as it provide more progress or has more neighbors at a low enough cost. We encompass the kingdom's worth by considering at each node of the conquering cost (C_n) + traveling cost(T_n) all divided by the number of neighbors(N_n) that are not already on our purchased list.

$$F(n) = \frac{C_n + T_n}{N_n}$$

In our case our algorithm will keep track of two different sets: conquered nodes(nodes that we actually traveled/conquered), surrendered nodes (nodes that we have gained through either conquering or surrendering).
Our correctness verifier will be taking in the surrendered list converting it into a set and then finding if the size if equal to number of nodes in our graph since we want to conquer everything.

We must consider a method to ensure that we are performing a tour and in the traveling purchaser problem, this problem is taken into consideration where the optimal solution requires for the walk taken to by a tour. It does this by connecting the last node of our walk with the starting node and then the algorithm only returns the minimum of all the possible paths that it came up with. Thus this intrinsically encourages the program to find a path that ends close to the starting node to minimize the final travel distance.