COMP 440 Homework 1

Tony Chen(xc12) and Adam Wang(sw33)

August 2016

1 Modeling sentence segmentation as search

- Suppose the sentence length is L, we define the state space model as following:
  - $S = s_0 \cup s_1 \cup s_2 \cup ... \cup s_L$, where, for $i = 0, 1, 2, ...L$, s_i represents the state that exactly i first characters have been segmented.
  - *Action* is defined as: for every word w in D that is a prefix from the i-th character in the sentence, $a_{i,w}$ is the action to jump to state $s_{i+len(w)}$.
  - $Successor(s_i, a_{i,w}) = s_{i+len(w)}$
  - $Cost(s_i, a_{i,w}) = 1$
  - $s_{start} = s_0$
  - $isGoal(s_i) = (s_i == s_L)$
- *Since the paths of unweighted graph of states are directed and never going "backward", there's no cycle in the graph.
  - Yes. BFS will be able to traverse the graph layer by layer from s_0 and once it reaches s_L, the current distance (number of layers traversed) is the minimum cost path. Time complexity will be $O(b^s)$ where b is the average number of neighbours and s is the number of layers from s_0 to s_L.
  - No. DFS will not guarantee the distance we have when we find s_L to be the minimum cost path.
  - Yes. UFS will be able to find a least cost path from s_0 to s_L in $O(L)$.
  - Yes. A* with a consistent heuristic will be able to find a least cost path from s_0 to s_L in $O(L)$.
  - Yes. Bellman-Ford will be able to find a least cost path from s_0 to s_L in $O(L^2|A|)$, where $|A|$ is the number of edges.
- Modify *Cost* function so that $Cost(s_i, a_{i,w}) = len(w) - 1$.
  BFS won't work now since it only works on unweighted graph. UFS, A*, and Bellman-Ford will still work since they all can generalize on weighted graph.
- Modify S so that for a state $s_{i,last}$ where $i > 0$, exactly i first characters have been segmented and the last segmented word is *last*.
  Modify *Cost* function so that $Cost(s_0, a_{0,w}) = 0$ and $Cost(s_{i,last}, a_{i,w}) = fluency(last, w)$.

2 Searchable Maps

- We define the cost as the time required to travel from s to t. Since we need a heuristic that never overestimates the cost, we define it as the lowest possible cost, which is obtained by a traveling along a straight line from s to t at the highest possible speed:

$$h(s,t) = \frac{G(s,t)}{S_H}$$

- The heuristic is defined as following:

$$h(s,t) = |T(s,L) - T(L,t)|$$

- Suppose the goal node is t, it is trivial that $h(t) = h_1(t) = h_2(t) = 0$. Consider any pair of node n and m where there is an action to get m from n. Suppose $h_1(n) \geq h_2(n)$, then $h(n) = h_1(n)$ and there are two possibilities: first, $h_1(m) \geq h_2(m)$, which makes $h(m) = h_1(m)$ and obviously makes h consistent on n and m (h is exactly h_1); second, $h_1(m) < h_2(m)$, then since $h(n) = h_1(n) \leq Cost(n,m) + h_1(m) < Cost(n,m) + h_2(m)$, h is also consistent on n and m. So h will always be consistent in this case
  Since h_1 and h_2 are symmetric, the above conclusion will also hold for $h_1(n) < h_2(n)$. So h is always consistent.

- According to part c, the basic idea is to take the max of all heuristics:

$$h(s,t) = max(|T(s,L_1) - T(L_1,t)|, |T(s,L_2) - T(L_2,t)|, ..., |T(s,L_K) - T(L_K,t)|, \frac{G(s,t)}{S_H})$$

- For adding edges, h will NOT remain consistent. Image the new edge draws a straight line from s to t, then as long as the original path estimate of h is not a straight line on its own, h will overestimate, which makes it not only inconsistent but also not admissible.
  For removing edges, h will still remain consistent. Since for any edge remains in the graph, none of the three parts in the inequality $h(m) \leq Cost(m,n) + h(n)$ will change. So h will remain consistent.

3 Designing Search Algorithms: Protein Folding

- S is composed by $s_{i,j,path}$'s which represents that the last "placed" residue has coordinates (i, j) and that the path before this residue is recorded by $path$, which is a list of coordinates. Note that this residue is the $(len(path) + 1)$-th one in the sequence.