AI report

Saurabh Daptardar, Srikanth Kuthuru

September 8, 2017

1 Problem 1

1. **State space model to segment a sentence into words:**
The entire sentence is considered at the beginning and is split into parts from the left giving rise to different possible states.
–**Start State:** Entire sentence.
–**Successor states and Cost:** Make a single split. Depending on the split location, it gives rise to different states. The example 'iseethecat' is used for illustration. Possible successors for this start state are 'i + seethecat' , 'is + eethecat', 'ise + ethecat' etc. Then, the first part of the splits (here they are 'i', 'is', 'ise') are tested for validity using a dictionary. So, here words like 'ise' are not valid and therefore they are terminated. This process continues, the second part of the split (here they are 'seethecat', 'eethecat', 'ethecat') is re-split again and the same validity test is done again. Each split gives rise to a state, and the edge cost is set to be 1.
– **Goal State:** If the second part of the split is a valid word, then it is a goal state.
2. **Search Algorithm:** We would use UCS to search the graph. The path with the minimum cost will be found by UCS. Path with minimum cost implies minimum splits are done across that path and therefore it has minimum words. UCS reaches the goal state first.
NOTE: Since, the edge costs equal to 1, UCS is same as BFS.
3. **For finding maximum split:** Set the edge cost to be -1. Use UCS to search the tree. UCS will give more priority to the smallest cost path. Path with maximum number of splits will have a more negative value. Hence, the optimal path will have maximum number of words.
NOTE: Since the search graph is a tree, negative edge costs do not pose any

problems.
4. **Finding most fluent sentence split:** Consider the same model as above. Let the edge costs for intitial splits (say level1 to level-2 of the tree) be 0. The next split will result in a total of 3 words, of which the first 2 words have to be valid. Let the fluency of these 2 words be f. Choose "-f" as the edge cost. For the next split, we have 4 words in which words:1,2,3 are valid. Here the edge cost will be the negative of fluency of words 2,3. This continues, until the goal state is reached.
Use UCS to search the tree. The path with most fluency will have the minimum total path cost and hence UCS can find that optimal split.

2 Problem 2

2.1 Part 1:

We need to define $h(n) \leq h^*(n)$ and it needs to be optimistic. So, for fastest path problem from node s to t (goal node) an admissible heuristic can be minimum time taken i.e. given a direct path from s to t traveling with maximum velocity i.e.

$$h(s) = \frac{G(s,t)}{S_H} \tag{1}$$

2.2 Part 2:

Given fastest paths or fastest travel time from all nodes to a landmark L , we need to formulate a consistent heuristic. Let g be the goal node then, we know from triangle inequality that:

$$T(n,L) \leq T(n,g) + T(g,L) \tag{2}$$
$$T(g,L) \leq T(n,g) + T(n,L) \tag{3}$$
$$\implies T(n,g) \geq |T(n,L) - T(g,L)| \tag{4}$$

So we can intuitively define our heuristic to be:

$$h_L(n) = |T(n,L) - T(g,L)|$$

To prove that this is consistent:
1. We can clearly see that $h_L(g) = 0$
2. We can clearly prove using triangle inequality that given neighbors n and n', $T(n,n') + h_L(n') \geq h_L(n)$

2.3 Part 3:

To prove maximum over heuristics preserves consistency or $h(s) = \max_i \ h_i(s)$ is consistent.
Let $h_1(s)$ and $h_2(s)$ be consistent heuristics, that implies

$$cost(n, n') + h_1(n') \geq h_1(n) \tag{5}$$
$$cost(n, n') + h_2(n') \geq h_2(n) \tag{6}$$

Now,

$$\begin{aligned} cost(n, n') + \max\ (h_1(n'), h_2(n')) &\geq cost(n, n') + h_1(n') && (7)\\ &\geq h_1(n) && (8)\end{aligned}$$

Similary,

$$\begin{aligned} cost(n, n') + \max\ (h_1(n'), h_2(n')) &\geq cost(n, n') + h_2(n') && (9)\\ &\geq h_2(n) && (10)\end{aligned}$$

From equations (8) and (10), it implies

$$cost(n, n') + max(h_1(n'), h_2(n')) \geq max(h_1(n), h_2(n)) \tag{11}$$

Hence proved that $h(s) = \max_i \ h_i(s)$ is consistent heuristic

2.4 Part 4

As shown in the part above, since maximum function preserves consistency, we can take maximum over multiple heuristics found in **Part 1** and **Part 2** to create one single better heuristic function

$$h_{max}(s) = \max\ \left(\frac{G(s,g)}{S_H}, h_{L_1}(s), h_{L_2}(s), \ldots, h_{L_K}(s)\right)$$

2.5 Part 5

Case 1: If edges are removed the cost between the nodes becomes infinite so h(n) still remains consistent Proof: Given $h(n)$ is consistent

$$\begin{aligned} cost(n, n') + h(n') &\geq h(n)\\ cost'(n, n') &\to \infty\\ cost(n, n') + h(n') = \infty &> h(n)\end{aligned}$$

So $h(n)$ still remains consistent

Case 2: If an edge is added it is not necessary that h(n) remains consistent. It depends on formulation of h(n) and cost of new edge
Proof by example:
Let $h(n) = 3$ and $h(n') = 2$, and min $cost(n, n') = 3$ (not direct path)
It can be shown for nodes that are not directly connected that consistency imples

$$\min\ cost(n, n') + h(n') \geq h(n)$$

If a edge from n to n' is added with $cost(n, n') = 0.5$, then $cost(n, n') + h(n') = 2.5$ is not $\geq h(n) = 3$
As seen from above example it depends on the edge cost being added.
Note: the heuristic in part 1 (for travel time) using Haversine distance will remain consistent (because no physical roads in given maximum speed limit will be faster than that).

3 Problem 3

3.1 Laying the ground work

- 1a. Code attached
- 1b. Code attached

3.2 Package Delivery

2 a. Formalizing the problem:
We formulate the state as a list of truck location and delivery status of each package. The truck location is (x, y) coordinate tuple and the delivery status is tuple of K elements (integers) as $(P_1, P_2, \ldots, P_K)$. Each delivery status $P_i\ \forall\ i \in \{1, 2, \ldots, K\}$ can take values $\{0, 1, 2\}$, where 0 means not yet picked up, 1 means picked and in transit, and 2 means delivered.

Start State: $[(T_x, T_y), (0, 0, \ldots, 0)]$
Goal State: $[(T_x, T_y), (2, 2, \ldots, 2)]$
Actions: N, S, E, W, Pickup if $P_i = 0$, Drop if $P_i = 1$
Note: pick up and drop only possible when delivery and pickup location match the current location

Cost: $1 + \sum_i I(P_i == 1)$, where $I(.)$ is identity function
Number of states for our problem: $m * n * 3^k$

2b. Code attached
2c. running $A*$ on DeliveryScenario1, number of states explored = 56
2d. running $A*$ on DeliveryScenario2, number of states explored = 34
2e. running $A*$ on DeliveryScenario3, number of states explored = 56

4 Problem 4

1. **State Space model:** The amino acid chain is grown stepwise. In each interation, an amino acid is added to the already existing chain of amino acids.
– **Start State:** First pair of amino acids.
– **Succesor states and Cost:** The next amino can have 3 positions to be at. It can either continue growing in the same direction as the previous link, or it turn 90 degress right or left. A checker is placed whcih makes sure that the new amino acid's location doesn't overlap with the already existing chain. We have closely followed the algorithm in [**?**]. Here, aggressive pruning is performed to control the size of the tree. Pruning depends on whether the amino acid being added is Polar/Hydrophobic. If it is hyperbhobic, and the new chain is going to have low energy then it is expanded with a certain probability 'P1'. If it is going to have high energy then it will be pruned with a probability 'P2'. If the new amino acid is Polar, then the no pruning is performed. These values 'P1' and 'P2' are tunable parameters. The cost of each edge is one.
– **Goal State:** If a protein has 'n' amino acids then the goal state is when thenth amino acid is added.

2. BFS or DFS can be used to find the minimum energy configurations in this case.

3. **Algorithm:** The above state space model is directly followed in the implementation. States are expanded according to the pruning probablities and the amino acids.

Pseudocode:
start = first pair of amino acids.

```
for loop - Keep adding amino acids one by one:
    Consider all locations of new amino acid. Use a checker function that doesn't consider the locations where the chain has already occupied.
    Calculate energies for each one of them.
    Z = mean(energies of proteins length k)
    if(new = Hydrophobic) then
        for all amino acid chains of length k that are generated:
            if(Energy $\geq$ Z): prune it with probablity P1
            else prune it with probability P2
Find minimum energy conformation in the final conformation list.
```

4. **Implementation:**

Our implementation is provided along with the other codes in the zipped folder. We have achieved a minimum energy state with energy 287.95 which is similar to the ones provided. Values of p1 = 0.8 and p2 = 0.65 are used. Lower values of P2 will take more time and will result in better configurations. This is not an optimal solution. Finding an optimal solution is NP-complete. Since, the tree grows exponentially it will take very large amont of time to find the optimal solution. The solution that this algorithm gives is better or comparable wilth Genetic or simulated annealing algorithms [as compared in paper [**?**]].