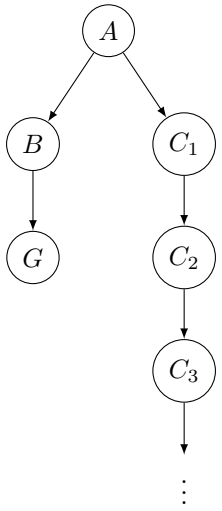
CS 540: Introduction to Artificial Intelligence
Homework Assignment # 4

Yien Xu

Question 1: Inadmissible Heuristic Affects Completeness [50 points]

We saw in the class that if the heuristic function h is inadmissible, A* search may find a suboptimal goal. We now show that A* may not even be complete, namely it may not terminate even if a goal with small cost exists. Consider the following graph where the right branch goes on forever: where A is the initial state



and G is the only goal state. Let

$$cost(A, B) = 1/2, \quad cost(B, G) = 1/2$$

so the solution path from A to G has a cost of 1. Furthermore, let

$$cost(A, C_1) = 1 \;\; cost(C_1, C_2) = 1/2 \;\; cost(C_2, C_3) = 1/4 \;\; cost(C_3, C_4) = 1/8 \;\; \ldots$$

so that the costs decrease by half toward the right.

Suppose for all states s the heuristic function $h(s) = 0$ *except* that $h(B) = 100$. This makes h inadmissible. Nonetheless, let us run the A* algorithm with this h.

1. (5 points) By definition, what range of $h(B)$ is considered admissible?

   If $h(B)$ is a lower bound of cost(B, G), then $h(B)$ is considered admissible. Therefore,

$$h(B) \in \left[0, \frac{1}{2}\right]$$

2. (20 points) Run the A* algorithm (on slide 21) by hand for five iterations: that is, you should execute step 5 five times. At the end of *each* iteration, show the following:

   - states in OPEN
   - states in CLOSED
   - for each state, show its f, g, h values
   - for each state, show its back pointer

   Function Values and Back Pointers format: State $S(g, h, f, pt)$.

   (a) Iteration 0:
   OPEN: $A(0, 0, 0, \text{Null})$
   CLOSED: EMPTY

   (b) Iteration 1:
   OPEN: $B\left(\frac{1}{2}, 100, 100 + \frac{1}{2}, A\right)$, $C_1\left(1, 0, 1, A\right)$
   CLOSED: $A(0, 0, 0, \text{Null})$

   (c) Iteration 2:
   OPEN: $B\left(\frac{1}{2}, 100, 100 + \frac{1}{2}, A\right)$, $C_2\left(\frac{3}{2}, 0, \frac{3}{2}, C_1\right)$
   CLOSED: $A(0, 0, 0, \text{Null})$, $C_1\left(1, 0, 1, A\right)$

   (d) Iteration 3:
   OPEN: $B\left(\frac{1}{2}, 100, 100 + \frac{1}{2}, A\right)$, $C_3\left(\frac{7}{4}, 0, \frac{7}{4}, C_2\right)$
   CLOSED: $A(0, 0, 0, \text{Null})$, $C_1\left(1, 0, 1, A\right)$, $C_2\left(\frac{3}{2}, 0, \frac{3}{2}, C_1\right)$

   (e) Iteration 4:
   OPEN: $B\left(\frac{1}{2}, 100, 100 + \frac{1}{2}, A\right)$, $C_4\left(\frac{15}{8}, 0, \frac{15}{8}, C_3\right)$
   CLOSED: $A(0, 0, 0, \text{Null})$, $C_1\left(1, 0, 1, A\right)$, $C_2\left(\frac{3}{2}, 0, \frac{3}{2}, C_1\right)$, $C_3\left(\frac{7}{4}, 0, \frac{7}{4}, C_2\right)$

   (f) Iteration 5:
   OPEN: $B\left(\frac{1}{2}, 100, 100 + \frac{1}{2}, A\right)$, $C_5\left(\frac{31}{16}, 0, \frac{31}{16}, C_4\right)$
   CLOSED: $A(0, 0, 0, \text{Null})$, $C_1\left(1, 0, 1, A\right)$, $C_2\left(\frac{3}{2}, 0, \frac{3}{2}, C_1\right)$, $C_3\left(\frac{7}{4}, 0, \frac{7}{4}, C_2\right)$, $C_4\left(\frac{15}{8}, 0, \frac{15}{8}, C_3\right)$

3. (5 points) What is $\lim_{i \to \infty} f(C_i)$? Show your derivation.

$$
\begin{aligned}
f(C_1) &= cost(A, C_1) = 1 \\
f(C_2) &= cost(A, C_1) + cost(C_1, C_2) = 1 + \frac{1}{2} \\
f(C_3) &= cost(A, C_1) + cost(C_1, C_2) + cost(C_2, C_3) = 1 + \frac{1}{2} + \frac{1}{4} \\
&\vdots \\
f(C_i) &= 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots + \frac{1}{2^{i-1}} \\
\lim_{i\to\infty} f(C_i) &= \sum_{k=0}^{\infty} \left(\frac{1}{2}\right)^k \\
&= \frac{1}{1-\frac{1}{2}} \\
&= 2
\end{aligned}
$$

4. (5 points) Explain in English why the search will not find G.

   As is shown above, $\lim_{i\to\infty} f(C_i) = 2$. So the values of $f = g + h$ of C_is are always less then $f(B) = 100 + \frac{1}{2}$. Therefore, B will never be popped out, and thus G cannot be found.

5. (10 points) Is there a range of $h(B)$ that is inadmissible but still allows A* search to find G? If yes, give the range; if no, explain why.

   If we want to reach the goal state G, we need to make sure that B can be popped out from OPEN first.

   $$f(B) = \frac{1}{2} + h(B) < \lim_{i\to\infty} f(C_i) = 2$$

   In addition, since we need $h(B)$ be inadmissible, we need to make sure that

   $$h(B) > \frac{1}{2}$$

   Now we get

   $$h(B) \in \left(\frac{1}{2}, \frac{3}{2}\right)$$

   Furthermore, we still want to make sure that the goal state gets popped out. After having B visited, we get

   $$f(G) = \frac{1}{2} + \frac{1}{2} + 0 = 1 < \lim_{i\to\infty} f(C_i) = 2$$

   This shows that after we confirmed that B can be popped out, G can be popped out for certain. In conclusion,

   $$h(B) \in \left(\frac{1}{2}, \frac{3}{2}\right).$$

6. (5 points) Is admissible h a sufficient condition, a necessary condition, both, or neither for A* search to find the optimal goal?

   Admissible h is a sufficient but not necessary condition for A* search to find the optimal goal. It is a sufficient because if we find correct admissible values for all nodes, we will definitely get the correct answer (mentioned in class). It is not a necessary condition because of the example in problem 5. Although we are guaranteed to find the optimal goal, the heuristic function we used was not admissible.

Question 2: Simulated Annealing [25 points]

Consider a state space consisting of integers. Suppose that the successors of a state x include all integers in $[x-10, x+10]$, including x itself. Consider the score function $f(x) = \max\{4-|x|, 2-|x-6|, 2-|x+6|\}$. It has three peaks with one forming a unique global maximum.

Recall that a random successor y is generated on each iteration of simulated annealing. If this successor is better than the current state x, it is always accepted. If it is not better, it will still be accepted with probability

$$p = \exp\left\{-\frac{|f(x)-f(y)|}{T}\right\}.$$

Recall that to do something with probability p, you generate a random number $z \in [0,1]$ uniformly and if $z \leq p$ you do it.

Use the temperature cooling scheme

$$T(i) = 2(0.9)^i,$$

where i is the iteration number. E.g., for the first iteration, the temperature will be $T(1) = 2(0.9)^1 = 1.8$.

Perform 8 iterations of simulated annealing using $x = 2$ as your starting point. We already "randomly" picked a successor y at each iteration for you, and also generated a "random number" z at that iteration for you to decide whether to go to an inferior successor, should you need it. These are listed in Table 1. Use these numbers. For each iteration, write down (1) the current point, (2) the temperature (rounded to the nearest thousandth), and (3) the probability of moving to the successor given the successor and the temperature.

Iteration Number	Random Successor y	Random Number z
1	3	0.102
2	1	0.223
3	1	0.504
4	4	0.493
5	2	0.312
6	3	0.508
7	4	0.982
8	3	0.887

Table 1: **Successor States and Random Numbers**

In [1]:
```
from __future__ import division
from __future__ import print_function
import numpy as np
from matplotlib import pyplot as plt
```

In [2]:
```
f = lambda x: max(4-np.abs(x), max(2-np.abs(x-6), 2-np.abs(x+6)))
T = lambda i: 2*(0.9**i)
p = lambda x, y, T: np.exp(-np.abs(f(x) - f(y)) / T)
```

In [3]:
```
succ = [3, 1, 1, 4, 2, 3, 4, 3]
random = [.102, .223, .504, .493, .312, .508, .982, .887]
x = 2
for i in xrange(8):
    y = succ[i]
    fy = f(y)
    fx = f(x)
    print("Iteration %d, current point %d, succ %d, Temp %.3f, " %(i+1, x, y, T(i+1)), end='')
    z = random[i]
    if fx < fy:
        x = y
        prob = 1
    else:
        prob = p(x, y, T(i+1))
        if z <= prob:
            x = y
    print("prob %.3f" %prob)
```

```
Iteration 1, current point 2, succ 3, Temp 1.800, prob 0.574
Iteration 2, current point 3, succ 1, Temp 1.620, prob 1.000
Iteration 3, current point 1, succ 1, Temp 1.458, prob 1.000
Iteration 4, current point 1, succ 4, Temp 1.312, prob 0.102
Iteration 5, current point 1, succ 2, Temp 1.181, prob 0.429
Iteration 6, current point 2, succ 3, Temp 1.063, prob 0.390
Iteration 7, current point 2, succ 4, Temp 0.957, prob 0.124
Iteration 8, current point 2, succ 3, Temp 0.861, prob 0.313
```

Question 3: City of Madison Open Data: Street Trees [25 points]

Visit the website `http://data-cityofmadison.opendata.arcgis.com`. Go City Datasets / SUSTAINABILITY to find the "Street Trees" dataset:

```
Street Trees  thumbnail
Shared by CityOfMadison
The different classifications of the tree data are as follows: ID:...
Custom License 9/11/2017 Spatial Dataset 112,511 Rows
```

Once you get into this dataset's webpage, you can zoom in to the map on top of the webpage to see where the trees are. Try find Computer Science Department and see the few trees around the building! There is a "Download" button on the upper right. Download the spreadsheet and take a look. You will see many trees with their attributes.

Suppose the city wants to inspect the trees for disease. An inspector starts from their office, drives a truck to visit each tree once, and goes back to the office at the end. The inspector wants to find the shortest route to do so. Because of traffic and one-way streets, the order the trees are visited affects the total distance the inspector needs to travel. It is therefore reasonable to represent a state by a permutation of the trees. Let there be n trees. For a particular permutation $t_1, \ldots, t_n$, the total distance of that state is defined as

$$f(t_1, \ldots, t_n) = d(O, t_1) + \sum_{i=1}^{n-1} d(t_i, t_{i+1}) + d(t_n, O)$$

where $d(a, b)$ is the driving distance from a to b, and O represents the office. One can view f as the score of the state, and the inspector wants to find a state with the minimum score.

1. How many states are there for n trees?

   The sequence of trees that we want to visit represents a state, e.g., $t_1, t_2, \ldots, t_n$. Therefore, by permutation, there are $n!$ states in total.

2. The inspector can perform hill climbing to approximately minimize the score. To do so, a successor (neighborhood) function needs to be defined for each state. It turns out that one valid way to generate successors for the state $t_1, \ldots, t_n$ is to pick a position $j \in [1, n-1]$ and swap t_j, t_{j+1}. (If you are interested: any permutation can be expressed as a product of adjacent transpositions.) What fraction of the state space does one neighborhood cover? Note for this problem, the neighborhood of a state does not include the state itself.

   First, we want to compute how many successors there are. As defined above, we only swap adjacent neighbors but excludes the current state, which leads to $\binom{n}{1} - 1 = n - 1$ successors. Therefore, the fraction is

   $$\frac{n-1}{n!}.$$

3. Find n from your Madison dataset and compute the number of states in scientific notation, round to one significant digit (i.e. $c \times 10^d$ where c is a single digit). This is why the inspector cannot enumerate all states!

   From the description of the data set, we see that $n = 112,511$. Using Stirling's approximation:

   $$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

   We can get

   $$112511! \approx 1 \times 10^{519455}$$

4. One way to estimate the worst case total distance is to assume that to travel between any two trees (or a tree and the office), the inspector has to travel the diameter of the city. Assume Madison has a diameter of 10 km. What is this worst case estimate of the total distance? Use the actual n for Madison and express your answer in LD, namely the average earth moon distance. Keep one significant digit.

   $$f(t_1, \ldots, t_{112511}) = \frac{10 + \sum_{i=1}^{112510} 10 + 10}{\text{LD} = 384402} = \frac{1125120}{384402} \approx 3\text{LD}$$

5. One way to estimate the best case total distance is to assume that to travel between any two trees (or a tree and the office), the inspector has to travel the minimum distance between any two trees. Assume that distance is 10 m. What is this best case estimate of the total distance? Express your answer in km.

$$f(t_1, \ldots, t_{112511}) = 0.01 + \sum_{i=1}^{112510} 0.01 + 0.01 = 1125.12 \text{ km}$$

6. Suppose the inspector drives at 25 miles per hour (note the Imperial unit). Can the inspector finish the job in one day?

$$1125.12 \div (25 \times 1.6) = 28.128 \text{ hrs} > 24 \text{ hrs}$$

So the inspector cannot finish the job in a day.