CSGA 6525 Artificial Intelligence

Fall 2017

Assignment #2

Minxia Ji

Question 1

(a)Define a state in this new problem

State: ([X, Y], $\begin{bmatrix} \mathrm{C\,C\,C} \\ \ldots\ldots\ldots \\ \mathrm{D\,C\,C} \end{bmatrix}$).

X stands for the ith row and Y stands for the ith column,[X, Y] stands for the current state. X belongs to [1,N] and Y belongs to [1,M]. The big following matrix stands for each room clean or dirty currently.

For example, state = ([2,3], $\begin{bmatrix} \mathrm{C\,C\,C} \\ \ldots\ldots\ldots \\ \mathrm{D\,C\,C} \end{bmatrix}$) means that the agent is currently in the room which is on the second row and third column of the grid and state[1,1] is dirty.

(b) Define the set of actions and the formal transition model for each

Suppose current state is [X^i, Y^i]

Actions: go left, go right, go up , go down, work

If action == go left and Yi >1: position changed to [Xi, Yi-1] else [X^i, Y^i]

If action == go right and Yi < M:position changed to [Xi, Yi+1] else [X^i, Y^i]

If action == go up and Xi < N:position changed to [Xi+1, Yi] else [X^i, Y^i]

If action == go down and Xi >1:position changed to [Xi-1, Yi] else [X^i, Y^i]

If [X^i, Y^i] is dirty, D, the vacuum-cleaner would start to work and change the corresponding current state in big following matrix from D/dirty to C/clean.

(c) Define all the necessary elements of a search problem

Initial state: any state as it satisfied the definition in question (a).

Goal state: a state with all elements = C/Clean in the big following matrix.

Actions and transitional model mentioned in question (b)

Heuristic: sometimes we need define the heuristic for a search problem.i.e., in A* search, heuristic is Manhattan distantce in vacuum problem: (X_goal - Xi)+(Y_goal - Yi)

(d) Explain using (c) how you would apply A* to address finding the shortest route to clean a single (known) dirty cell (xd,yd) avoiding any other other dirty rooms on the way.

From initial state, we can take up to 4 actions. The f(n) = g(n)+h(n), g(n) means the cost to reach this state from initial state and h(n) means the cost to reach (xd,yd) from this state. Based on the concept of A* search, we will always move to the state with smallest f(n) until we reach (xd,yd).
In order to avoid any other dirty rooms on the way, we should add an if condition: **if** next state is D/dirty and is not the goal state, f(n) = g(n)+h(n) **+ a very large number**(as a punishment) **else** f(n) = g(n)+h(n). Then according to A* search heuristic, our agent will never go to that dirty state.

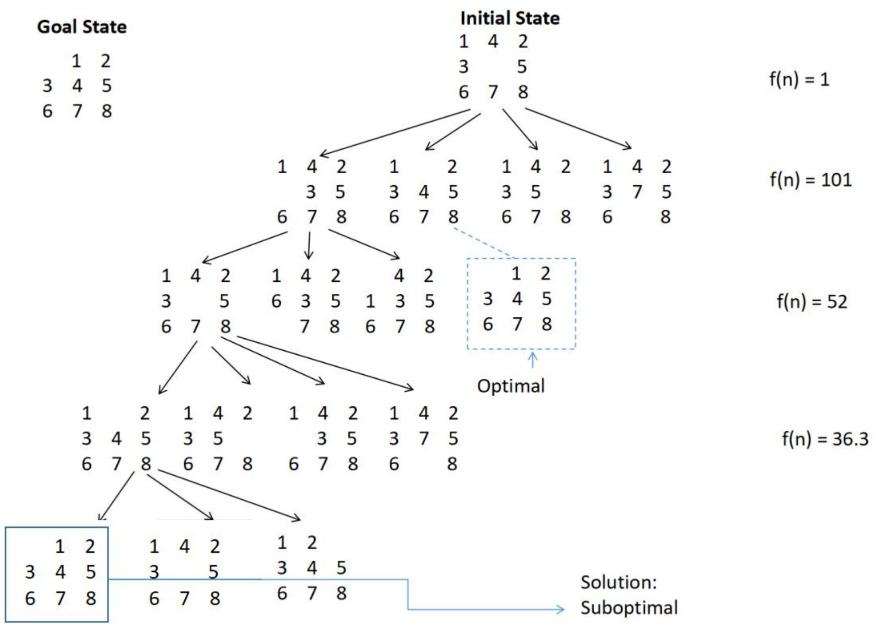
Question 2

(a) Invent a heuristic function for the 8-puzzle that sometimes overestimates (i.e., it is not admissible).

Heuristic: h(s0) = 1 for initial state, otherwise h(s) = 100/cost. Cost stands for the cost so far from the initial state to current state. Therefore, all successors at same level would have same value of evaluation function (f(s)=g(s)+h(s)). At beginning, A* search will do depth first search.

(b) Demonstrate (by example or mathematically) how this can sometimes lead to a suboptimal solution.

A* search tree might look like this

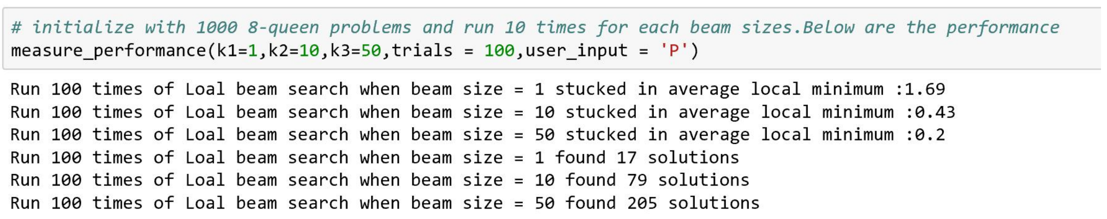


In this example, following the heuristic we find a solution marked with an arrow. That is a **suboptimal** solution since we should have found the optimal solution which is: move '4' down then move '1' to right and we reach the goal state.

Question 3

Question (a),(b) are in .py file

(c) Generate results that show the performance of each value of k for many examples and explain those results.



```
# initialize with 1000 8-queen problems and run 10 times for each beam sizes.Below are the performance
measure_performance(k1=1,k2=10,k3=50,trials = 100,user_input = 'P')
```

```
Run 100 times of Loal beam search when beam size = 1 stucked in average local minimum :1.69
Run 100 times of Loal beam search when beam size = 10 stucked in average local minimum :0.43
Run 100 times of Loal beam search when beam size = 50 stucked in average local minimum :0.2
Run 100 times of Loal beam search when beam size = 1 found 17 solutions
Run 100 times of Loal beam search when beam size = 10 found 79 solutions
Run 100 times of Loal beam search when beam size = 50 found 205 solutions
```

When we generated 100 trials, the performance with different Ks are good since we could find solutions in either value of k. With the increasing of the value of k, the average of local minimum became smaller and we found solutions for more times. Therefore, we can conclude that when the beam **size get larger**, the local beam search will perform **better**.

Quesiton 4

(a) How minimax works for regular, two player, zero sum games?

Explaination:For zero sum game, one win means the other one lose.suppose we have 2 players, say A and B. Here, we choose A's utility as our heuristic throughout the whole game.If A is the one who makes first move, A would choose the state which would maximize its utility. We call this step 'MAX'. Then it is B's turn to make move, B's goal is to minimize A's utility. We call this step 'MIN'. 'MAX' A > 'MIN' A > 'MAX' A > 'MIN' A...Finally we can reach our terminal point: either A or B win the game.

(b)How to modify minimax to work for two player non-zero sum games?

Explaination: There are several types of non-zero sum games. If in each move, the player would get positive or negative utility, then we don't need to modify the minimax. Suppose we have a player A and B. A took first move, A would choose the largest positive utility. Now it's B's turn, B would choose the largest positive utility for B, which means B would choose the largest negative utility for A. Thus we remain our minimax unchanged: 'MAX' A > 'MIN' A > 'MAX' A > 'MIN' A...until A and B reach the terminal state.

If A and B are always get positive utilities, the we should modify our minimax to alwaysmax: 'MAX' A > 'MAX' B> 'MAX' A...until A and B reach the terminal state.

(c)How to modify minimax for multi-player, non-zero sum games?

If we have multiple players say A, B ,C ..., we should modify our minimax to alwaysmax: 'MAX' A > 'MAX' B> 'MAX' C...until A, B, C ... reach the terminal state. Why we should always do max for each player? Because from a comprehensive view, even in some cases, some players try to minimize other player/players, in the end this behavior equals to maximize themselves.