CS440 Project 3: Probabilistic Search and Destroy

Achintya Singh, Tapan Patel

April 10, 2021

Exercise 1

```
    def updateBelief(self,r,c):
        tt=self.board.board[r][c]
        self.belief[r][c]*=terrain_diff[tt]
        if self.probReset: # rescale whole prob. matrix
            total=sum(sum(self.belief,[]))
            self.belief=[[c/total for c in r] for r in self.belief]
```

Updating the belief state after a failure, the cell(j), which was searched, is updated as

$$P(j)|t+1 <= P(j)|t \cdot \text{False\_Negative\_Rate(j)}$$

However, the belief state of the rest of the cells is updated in a different manner which is equivalent to normalization

$$P(i)|t+1 <= P(i)|t \cdot (1/(\text{TotalProb - reduction}))$$

where reduction is the delta of P(j) from t to t+1, derived from eq6 as

$$P(j)|t - P(j)|t+1 => P(j)|t \cdot (1\text{-False\_Negative\_Rate(j)})$$

This can also be viewed as a re-scaling of probabilities matrix with updated failure cell, and setting it to 0 to 1.

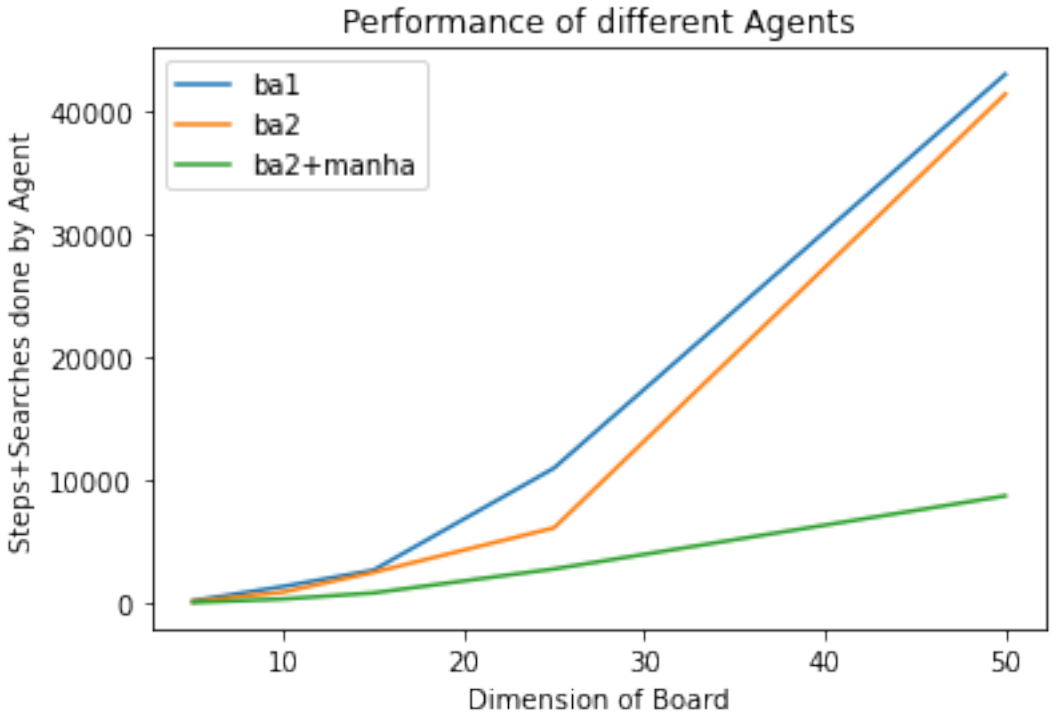
Exercise 2

The probability of finding a target in the cell can be given as

P(Target found in Cell I | Observations T) <= Belief_P(Cell I | Observations T)·(1–False_Negative_Rate(I))

Exercise 3

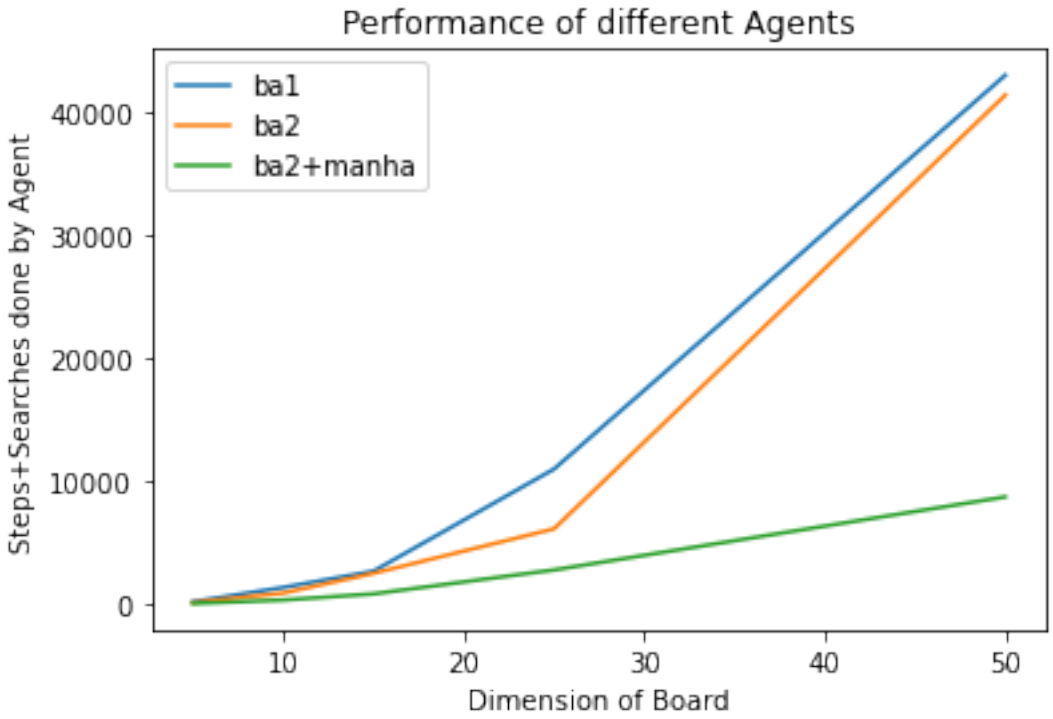
The Basic Agent 1 was implemented using *eq6* and Basic Agent 2 was implemented using *eq8*. Below is a plotted graph of those agents performance vs various Board dimensions.



For smaller board sizes (<10), Basic Agent 1 was better and later on Basic Agent 2 gets better on average. Note, for both of the the agents, the re-scaling was applied after every update of failure in search.

```
def bestMove(self):
manha=lambda ir,ic:abs(self.current_pos[0]-ir)+abs(self.current_pos[1]-ic)
if self.agent_version=="ba1":
    getprob=lambda ir,ic:self.belief[ir][ic]
elif self.agent_version=="ba2":
    getprob_=lambda
        ir,ic:self.belief[ir][ic]*(1-terrain_diff[self.board.board[ir][ic]])
    inner=[[getprob_(ir,ic) for ic,v in enumerate(r)] for ir,r in
        enumerate(self.belief)]
    innertotal=sum(sum(inner,[]))
    inner=[[v/innertotal for v in r] for r in inner]
    getprob=lambda ir,ic:inner[ir][ic]
```

Exercise 4



The improved Agent leverages the Manhattan distance while choosing the next cell on top of Basic Agent 2, that is probability of finding the target in a given cell. As seen in the plot, it performs better than both the basic agents. Reason being, it respects both of the measures in the performance metrics (i.e. reducing the distance traveled by including the Manhattan distance and reducing the number of searches by including the probability of finding the target).

```
    elif self.agent_version=="ba2+manha":
        getprob_=lambda
            ir,ic:self.belief[ir][ic]*(1-terrain_diff[self.board.board[ir][ic]])/(1+manha(ir,ic))
        inner=[[getprob_(ir,ic) for ic,v in enumerate(r)] for ir,r in
            enumerate(self.belief)]
        innertotal=sum(sum(inner,[]))
        inner=[[v/innertotal for v in r] for r in inner]
        getprob=lambda ir,ic:inner[ir][ic]
```