Group Member:
Xin Yang(xy213)
Zhuohang Li(zl299)

CS 520: Assignment 3 - Probabilistic Search (and Destroy)

1. A Stationary Target

1) According to the Bayes' Theorem, we can have the following equation: P (Target in Cell i |Observations t ∧ Failure in Cell j)=P (Target in Cell i ^ Observations t ∧ Failure in Cell j)/P(Observations t ∧ Failure in Cell j). The P(Observations t ∧ Failure in Cell j) can be calculated by the opposite, which is 1 minus the probability of success in Cell j. Define the belief of target in Cell j is Bj, the false negative rate is N, we can get the P(Observations t ∧ Failure in Cell j) to be 1-Bj(1-N). The belief of target in Cell i still remain the same as if we expand the P (Target in Cell i ^ Observations t ∧ Failure in Cell j), we will find that it equals to P(Target in Cell i)*P(Observations t ∧ Failure in Cell j | Target in Cell i). Since when the target exists in Cell i, the Observation t and Failure in Cell j must have happened, the P(Observations t ∧ Failure in Cell j | Target in Cell i) is 1. If the Cell i is the same as Cell j, then the probability is Bj*N. After each iteration, we normalize the global beliefs for the next step.

2) The difference between target is found in Cell i and target is in Cell i is that there is a false negative rate N. The probability of finding the target in Cell i is the probability of Cell i containing target times the possibility that Cell i shows the target directly. P(Found in Cell i) = P(Cell i contains target) * P(Cell i shows the target directly), which is Bi*N.

3) Our program updates the beliefs of all cells, then go through the whole board to find the cells with the highest probability in terms of the rules we set. If there exist multiple highest preferred cells, our strategy is to keep all such cells in a list, and randomly choose one.

We set our experiment on a fixed map and run 1000 iterations. Each iteration we use rule1 and rule2 to search the target and randomly move the target to a new location before the next iteration. The result shows the average number of search is 5831.504 using rule 1 and 4772.103 using rule 2. Then we try to run on different maps. It turns out that rule 2 is usually 1-2k steps fewer than rule 1. Our implementation shows that on average rule 2 has better performance. We think this is mainly because rule 2 is taking the probability of a false navigation into considering. So it will start with cells that highest probability of finding the target and therefore could finish earlier than rule 1 which would stuck on those cells with high false navigation probabilities.

4) Under this constraint, we need to combine the previously calculated probability and the distance information, the balance of this two information determines the efficiency of the algorithm. What we want to achieve is to keep the original belief as possible and adding the distance factor to adjust the weight. The farther distance should derive a lower updated belief. Our approach is to set a shrink factor, here is 0.999. We set the updated belief to be the original belief B times the shrink factor to the power of distance/4. The 4 here is to balance the effect of shrink factor.

   Experiment results are as follow:
   For rule 1, the naive decision rule has an average of 6959.87 searches and 231891.22 motions while our distance-based decision rule has an average of 4978.7 searches and 14507.86 motions.
   For rule 2, the naive decision rule has an average of 4921.0 searches and 164165.27 motions while our distance-based decision rule has an average of 4434.96 searches and 13244.88 motions.
   We noticed that bringing distance into consideration can sometimes result in a slight increment of the number of search operations. But as we can see from the result, our distance-based rule can reduce the number of motion operations to one-tenth of the naive decision rule. Since the number of

motions is huge comparing to searches, the overall performance is significantly better.

5) The balance between the belief and the distance decides the overall strategy. Our implementation can change the shrink factor and the denominator related to the distance to decide if we are going to search our target in a brighter place, or just head towards the most possible place.

2. A Moving Target

1) Since we can get the type of cells that target is transferring, every time before we get the best cell to search, we update the belief according to the type information. Our idea is to use the belief of the previous state and the transition model to update the belief of the current state. Assume the target was seen at a type1-type2 border. It means that if the target was at a cell of type 1 last state, it should now be at a neighbor cell of type 2 or the other way around. This could be model as the belief of a type 1 cell should transfer to a neighbor cell of type 2, and the belief of a type 2 cell should transfer to a neighbor cell of type 1, divide equally if there are more than one such cells. The implementation details are as follow: If the cell has an irrelevant type, then we set the belief as 0 to get rid of our consideration. Then we update the belief according to the neighbours. For each prime cell that meets the type, we update all its qualified neighbours' beliefs to be the neighbour cell's belief + prime cell's belief / number of such neighbours. This will make sure the cell gets higher belief when more prime cells have such a qualified neighbour.

   This belief update will discard lots of useless cells, thus making the search easier and more efficient.

   We use a fixed map and run rule 1 and rule 2 each for 1000 times. Average steps of rule 1 is 36.205, average steps of rule 2 is 35.278. Rule 2 is still better and we think the reason is the same as we've discussed in 1.3.

// re-do q4
We use the same strategy to add the distance information as the one in stationary part.

Experiment results are as follow:
For rule 1, the naive decision rule has an average of 35.036 searches and 950.543 motions while our distance-based decision rule has an average of 35.076 searches and 913.036 motions.
For rule 2, the naive decision rule has an average of 35.097 searches and 936.561 motions while our distance-based decision rule has an average of 33.749 searches and 859.546 motions.
We notice that sometimes the number of searches and motions of our algorithm can be larger than that of the naive decision rule which never happends in question 4). This might because the extra piece provided has greatly narrowed down the number of search cells. Therefore the distribution of search cell becomes sparse and the number of reachable cells of our distanced-based algorithm reduces. That's why in extreme case it might use more searches and more motions that the other agent who is thinking globally by following naive rule. But on average the distance-based algorithm still has a better performance.