Report -Assignment 3- Probabilistic Search (and Destroy)

Sharvani Pratinidhi (NetID: spp133)
Amit Patil (NetID: amp508)
Pranita Eugena Burani (NetID: peb63)

- Map Size: 50 x 50
- Every Map contains:
  20% Flat cells- with false negative rate – 0.1
  30% Hilly cells- with false negative rate – 0.3
  30% Forest cells- with false negative rate – 0.7
  20% Cave cells- with false negative rate – 0.9
- Target is placed randomly and can be in area of the map

A Stationary Target

1) *Given observations up to time t (Observations$_t$), and a failure searching Cell$_j$ (Observations$_{t+1}$ = Observations$_t$ ∧ Failure in Cell$_j$), how can Bayes' theorem be used to efficiently update the belief state, i.e., compute: P (Target in Cell$_i$ / Observations$_t$ ∧ Failure in Cell$_j$).*

   50 x 50 Belief Matrix with belief of target in each cell is given as

$$Belief[Cell_i] = P(Target\ in\ Cell_i\ |Observations\ through\ time\ t)$$

(at time t=0) $Belief[Cell_i] = P(Target\ in\ Cell_i) = {}^{1}/_{No.of\ cells} = \frac{1}{2500}$

On failure of finding a target in $Cell_j$, we update our belief matrix as follows:

$$Belief[Cell_i] = P(Target\ in\ Cell_i\ |Observations\ through\ time\ \hat{}failure\ in\ Cell_j)$$

$$Belief_{it} = P(T_i|X_t)$$

$$\Delta = P(S_j = Success|T_i, X_t)P(T_i|X_t) = \left(1 - P(S_j = Failure|T_i, X_t)\right)P(T_i|X_t)$$

$$\alpha = \frac{1}{P(S_j = Failure|T_j, X_t)P(T_j|X_t) + \sum_{k\neq j} P(S_j = Failure|T_k, X_t)P(T_k|X_t)}$$

$P(S_j = Failure | T_k, X_t) =$ 1 , since you can never find the target if the target in somewhere else

$$P(T_i | S_j = Failure, X_t) = \begin{cases} \alpha\ (Belief_{it} - \Delta) & i = j \\ \alpha\ Belief_{it} & i \neq j \end{cases}$$

Where

$Belief_{it}$: $P(Target\ in\ Cell_i\ |Observations\ through\ time\ t)$

S_j : Result of Search in $Cell_j$

T_i : Target in $Cell_i$

X_t : Observations till time t

2) *Given the observations up to time t, the belief state captures the current probability the target is in a given cell. What is the probability that the target will be found in* $Cell_i$ *if it is searched:*

$$P(Target\ found\ in\ Cell_i | Observations_t) = P(Target\ found\ in\ Cell_i | X_t)$$
$$= P(T_i, S_i = Success | X_t)$$

$$P(T_i, S_i = Success | X_t) = P(S_i = Success | T_i) P(T_i | X_t) = (1 - P(S_i = Failure | T_i)) Belief_{it}$$

Success rate depends on the terrain type, doesn't change with time

3) *Consider comparing the following two decision rules:*

– Rule 1: At any time, search the cell with the highest probability of containing the target.

– Rule 2: At any time, search the cell with the highest probability of finding the target.

For either rule, in the case of ties between cells, consider breaking ties arbitrarily. How can these rules be interpreted / implemented in terms of the known probabilities and belief states?

For a fixed map, consider repeatedly using each rule to locate the target (replacing the target at a new, uniformly chosen location each time it is discovered). On average, which performs better (i.e., requires less searches), Rule 1 or Rule 2? Why do you think that is? Does that hold across multiple maps?

The solver will recursively choose the cell with the best value based on Rule 1 or Rule 2.

Rule 1: $Belief_{it} = P(T_i | X_t)$

Rule 2: $(1 - P(S_i = Failure | T_i)) Belief_{it}$

The knowledge of the solver consists the prior beliefs $P(T_k | X_t)$ of all cells and the false negative rate $P(S_i = Failure | T_i, X_t)$ of searching a certain type of terrain. The false negative rates do not change through time whereas the belief is stored as a 50-by-50 belief matrix. Whenever the search is failed, this belief matrix is updated (implemented in function *update_belief*)

$$P(T_k|X_{t+1}) = \begin{cases} \alpha\,(Belief_{it} - \Delta) & i = j \\ \alpha\,Belief_{it} & i \neq j \end{cases}$$

Each round, the solver finds a best cell to search based on either Rule 1 or Rule 2, then after getting a response from Terrain Map, it will update belief matrix for the decision in the next round.

To compare the performance of Rule 1 and Rule 2, we placed the target in all four type of target terrain. As a result, there are 4 terrains × 2 rules = 8 cases, we did 100 rounds of search for each case. The average number of steps needed to find the target is listed in the Table below.

Comparison of Rule 1 and Rule 2 using Average number of steps needed to find the target as a performance parameter

Type	Rule 1	Rule 2
Target in Flat	3127.78	1179.19
Target in Hilly	4545.92	2678.17
Target in Forested	9548.81	5227.85
Target in Cave	10717.09	13832.46
Average	6984.9	5729.418

Result:

Rule 2 performs better compared to Rule 1 in the cases where Target is in Flat, Hilly regions, and slightly better in case of Forested regions. But in the case where target is in cave Rule 1 performs better than Rule 2

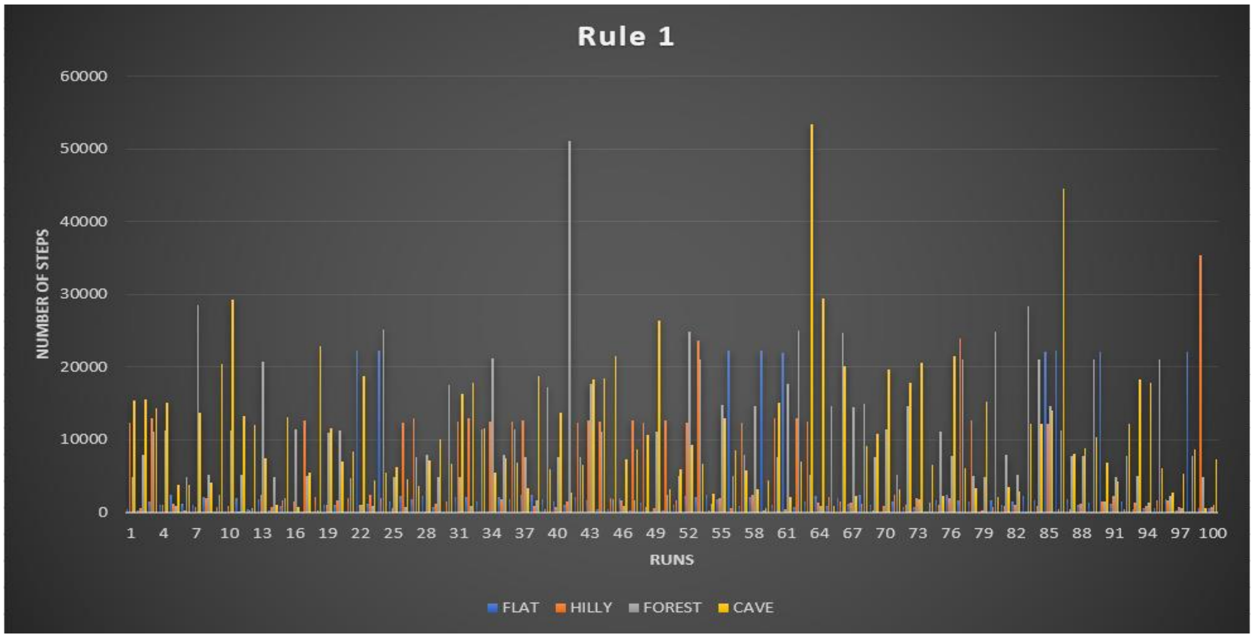
Reason:

This is because Rule 2 along with the belief it considers the probability of finding the target in a given terrain, whereas Rule 1 ignores this and gives importance to all types of cells equally. Rule 2 first searches Flat then Hilly followed by Forested regions then comes to caves, this is the reason why it performs bad in case of caves and very fast in case of Flat cells.

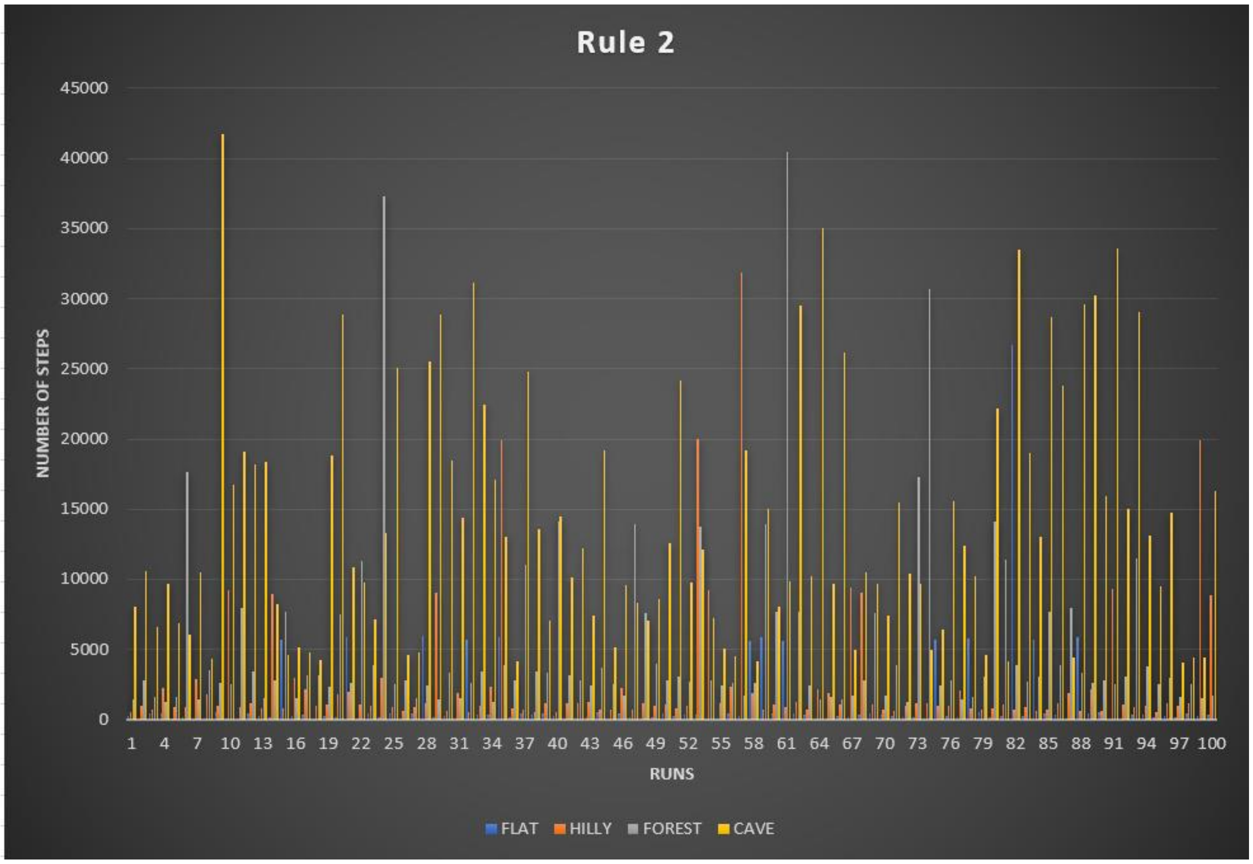
This behavior across multiple maps because the solver makes decisions based on the belief of the cell and the terrain type, it is not influenced by location of target in a particular terrain. In other words, changing the map does not change the fraction of each terrain. Furthermore, the target is set randomly among the cells with the same terrain, and the agent also search randomly among the cells with the best probability, this confirms that the order where the Target is placed in the same terrain cells is irrelevant to this problem.

Graphs:

Rule1:



Rule2:



4) *Consider modifying the problem in the following way: at any time, you may only search the cell at your current location or move to a neighboring cell (up/down, left/right). Search or motion each constitute a single 'action'. In this case, the 'best' cell to search by the previous rules may be out of reach and require travel. One possibility is to simply move to the cell indicated by the previous rules and search it, but this may incur a large cost in terms of required travel. How can you use the belief state and your current location to determine whether to search or move (and where to move), and minimize the total number of actions required? Derive a decision rule based on the current belief state and current location and compare its performance to the rule of simply always traveling to the next cell indicated by Rule 1 or Rule 2. Discuss.*

Here we are considering two parameters while selecting the next cell. The first is the current belief of a cell and the second is the Distance of the current cell from the target cell. We are selecting the cell which has maximum value for (belief of the cell/distance from the target). While calculating the distance of the current cell from the target cell we are using the Manhattan Distance – abs(x_target – x_cell) + abs(y_target – y_cell).

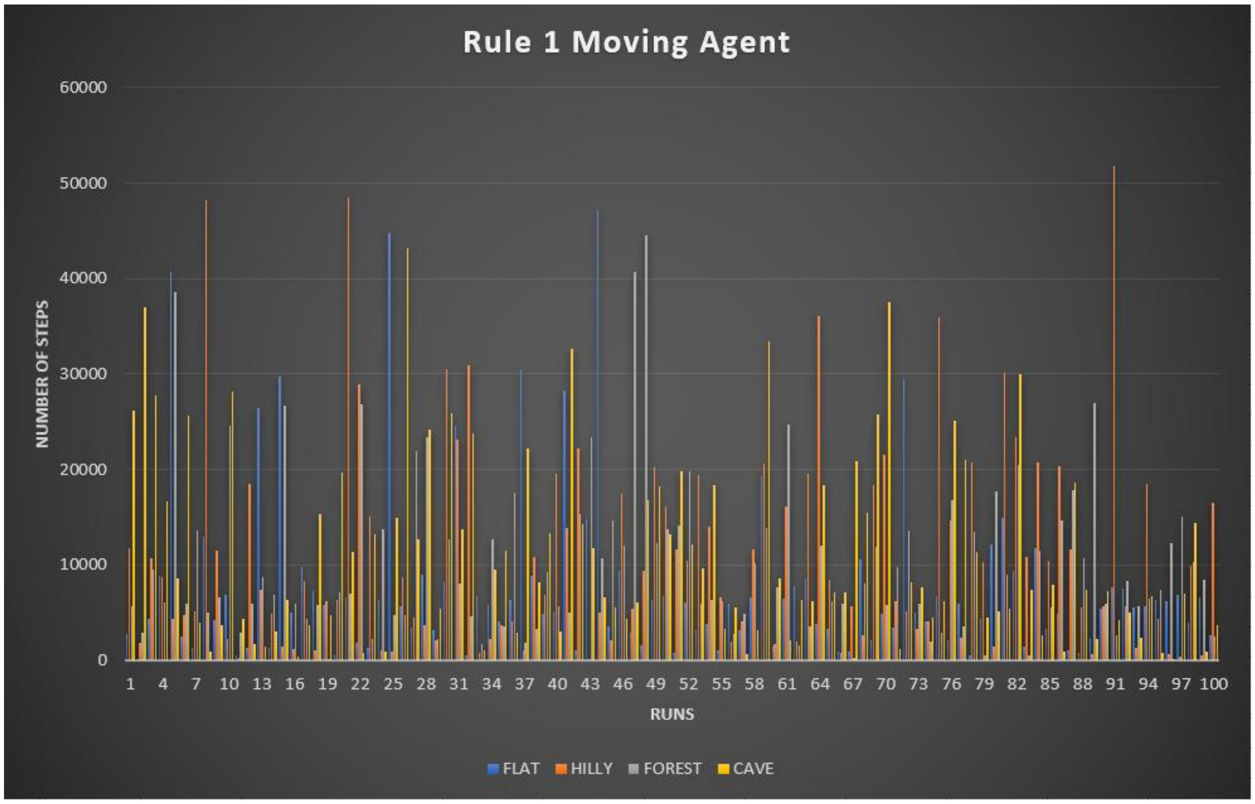
We measured the average number of steps over 100 iterations and got this result with these new modification for Rule 1 and Rule 2.

Type	Rule 1	Rule 2
Target in Flat	7692.88	770.92
Target in Hilly	11092.31	2219.86
Target in Forested	10320.89	6681.15
Target in Cave	11214.44	12356.94
Average	10080.13	5507.21

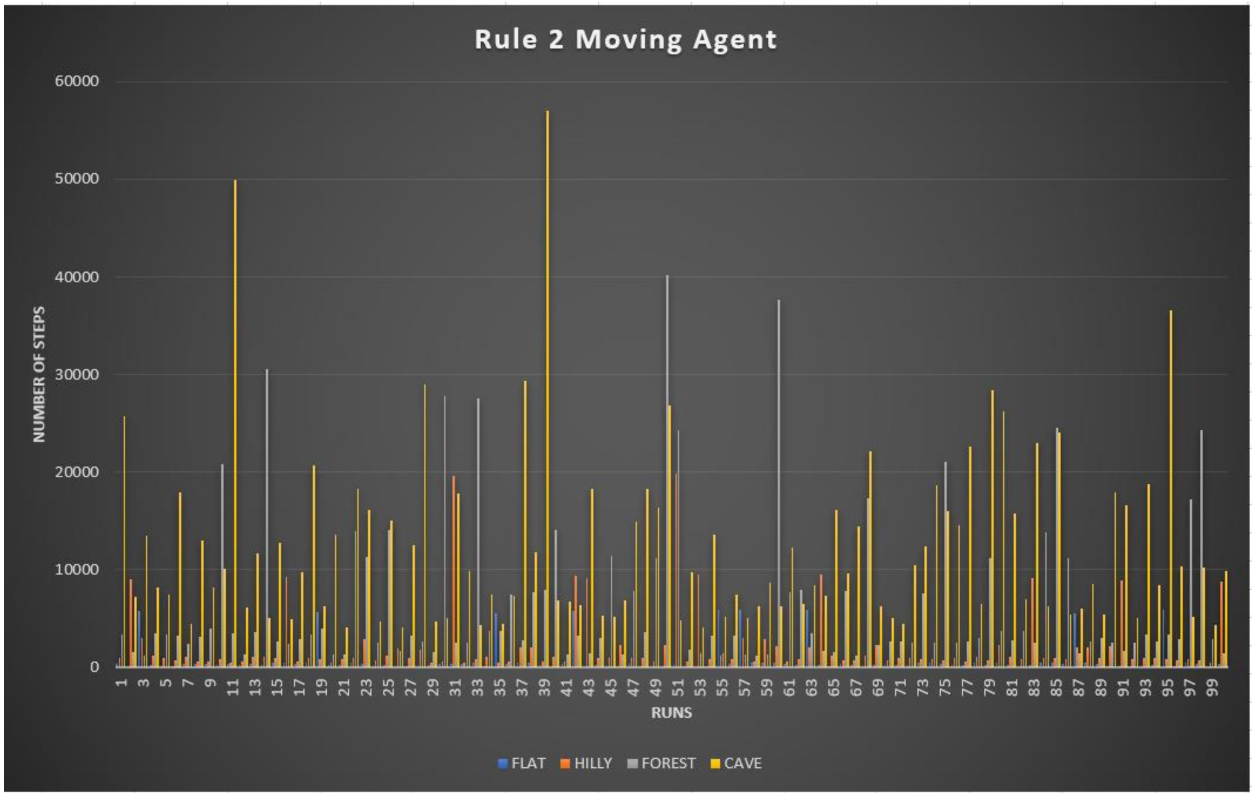
As we can see here that the Rule2 is performing better than Rule1 but the average steps taken by this method for both the rules are greater compared to the previous method. This is expected result because in this method most of the times we are we are moving to the neighboring cells. The movement of the agent is restricted by the only to the limited distance due to the cost of moving.

Graphs:

Rule1:



Rule2:



5) An old joke goes something like the following:
A policeman sees a drunk man searching for something under a streetlight and asks what the drunk has lost. He says he lost his keys and they both look under the streetlight together. After a few minutes the policeman asks if he is sure he lost them here, and the drunk replies, no, and that he lost them in the park. The policeman asks why he is searching here, and the drunk replies, "the light is better here".
In light of the results of this project, discuss.

To Discuss the above example in the light of the results of the project

- Drunk Man/Policeman – Hunter
- Lost Keys – Target
- Flat – Under the street light
- Hilly – Around the street light
- Forested – Area nearer to the Park than to the street light
- Cave – Park

Assumptions.
Probability of losing the keys in the park is large (from the drunk man's answer / belief) and decreases as we move towards the street light
Probability of Finding the key if it is lost under the street light is large and decreases as we move towards the park ("*the light is better here (street light)")*

So basically the drunk man is using Rule 2 to find the keys, If the keys is not lost in the park, then what the drunk man is doing is right, but given that probability of losing the keys in park is higher, drunk man needs to search for a longer time and should gradually consider searching on the way towards the park and in the park.

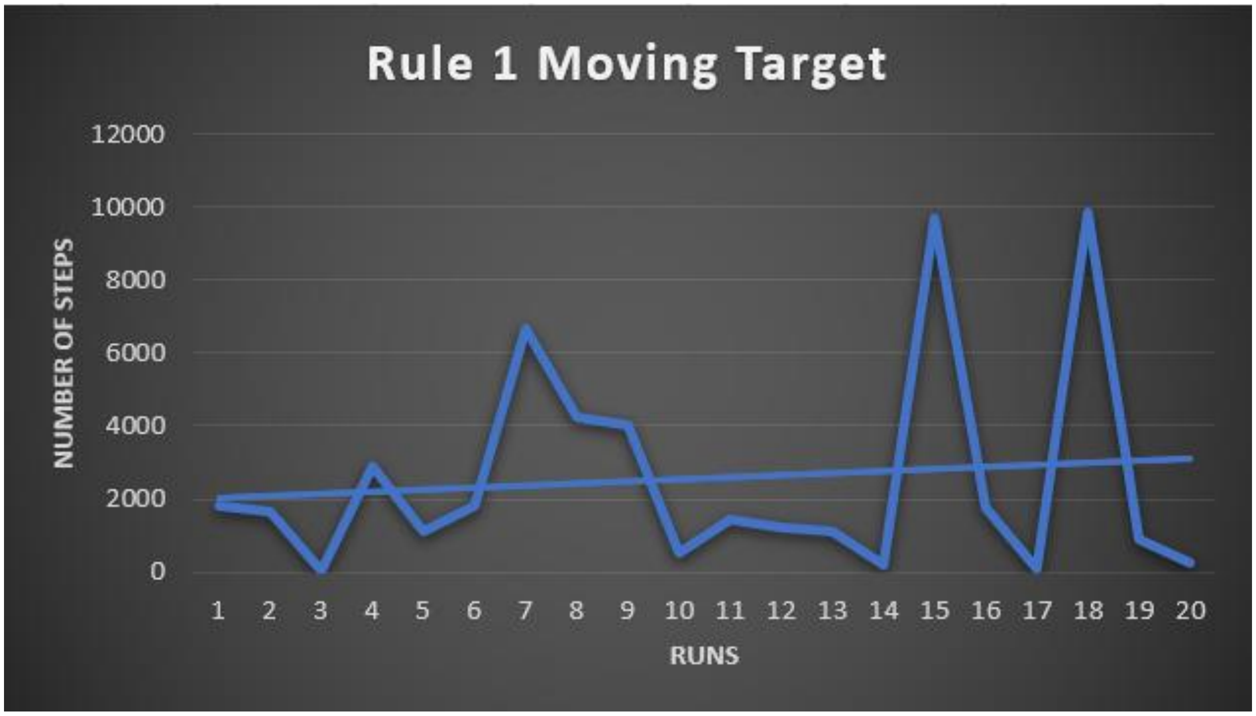
Q2. Moving Target

In this case, after each failed attempt to search the target, the target moves to the neighboring cells and surveillance reports gives the information about the border type1 and type2 between which the target is moved. We mark these cells in the map having type = type1 or type =type2. So at any time t we need to only check those cells which are marked as type1 or type2 cells. After we have this information we make the relative probabilities for cells of type3 and type4 as 0. And the initial belief of this cells will be distributed to the cells which are marked as type1 or type2 cells. And for selecting the next cell we are choosing the cell with highest probability value In belief matrix.

We are updating belief for type1 and type2 cells as follows:
Belief(I,j) = belief(I,j) * (1+(prob that are made 0|cells that are not zero))

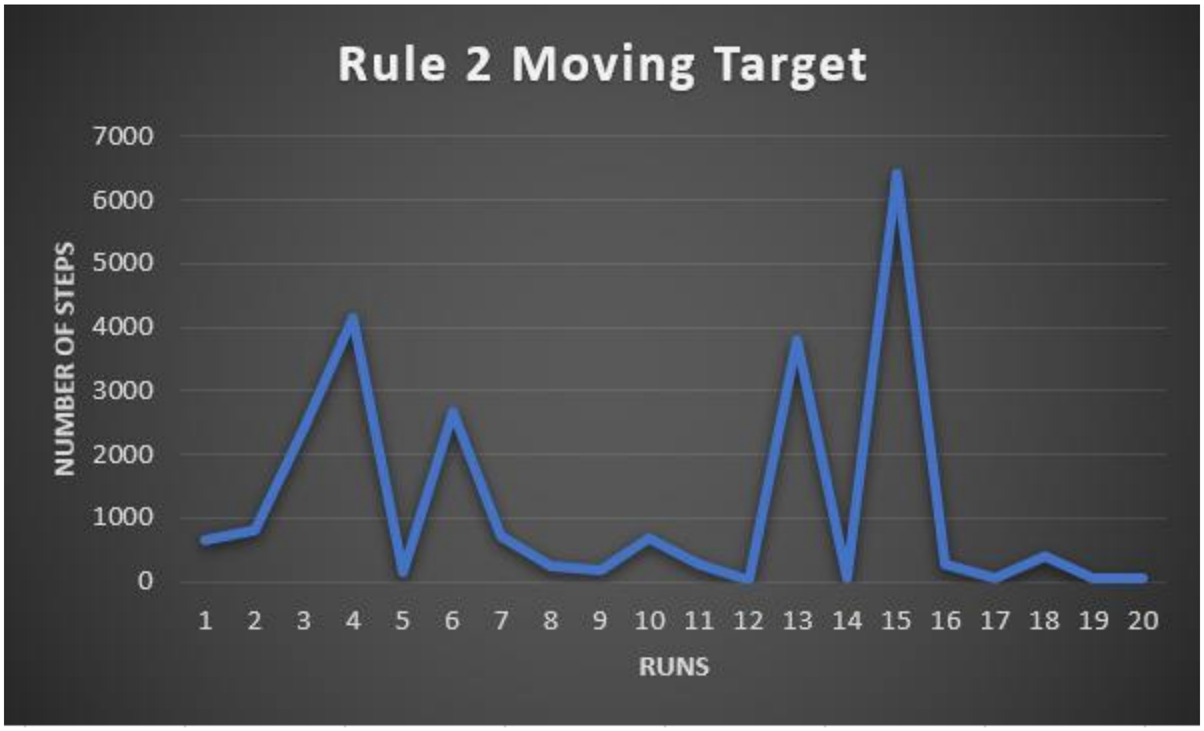
We executed this program for 20 times and got these results:

Rule1:



Average number of search moves – 2561

Rule2:

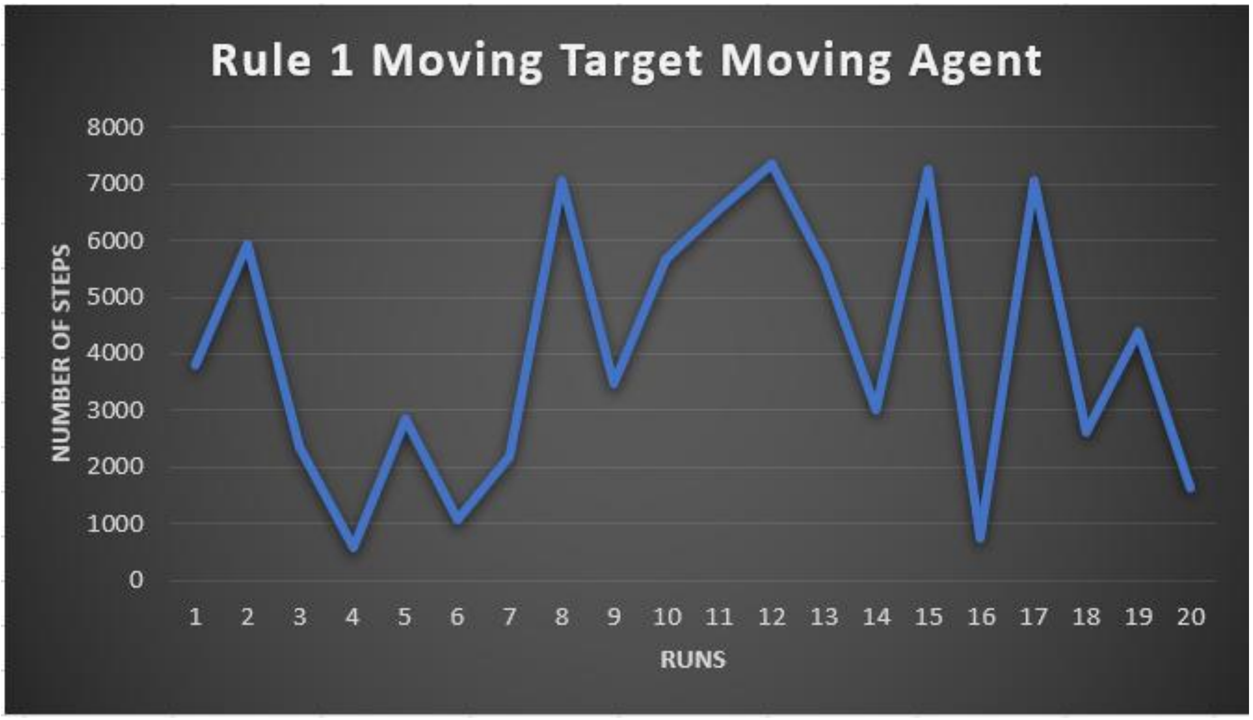


Average number of search moves – 1201.7

As we can see that the Rule2 is performing better than Rule1.

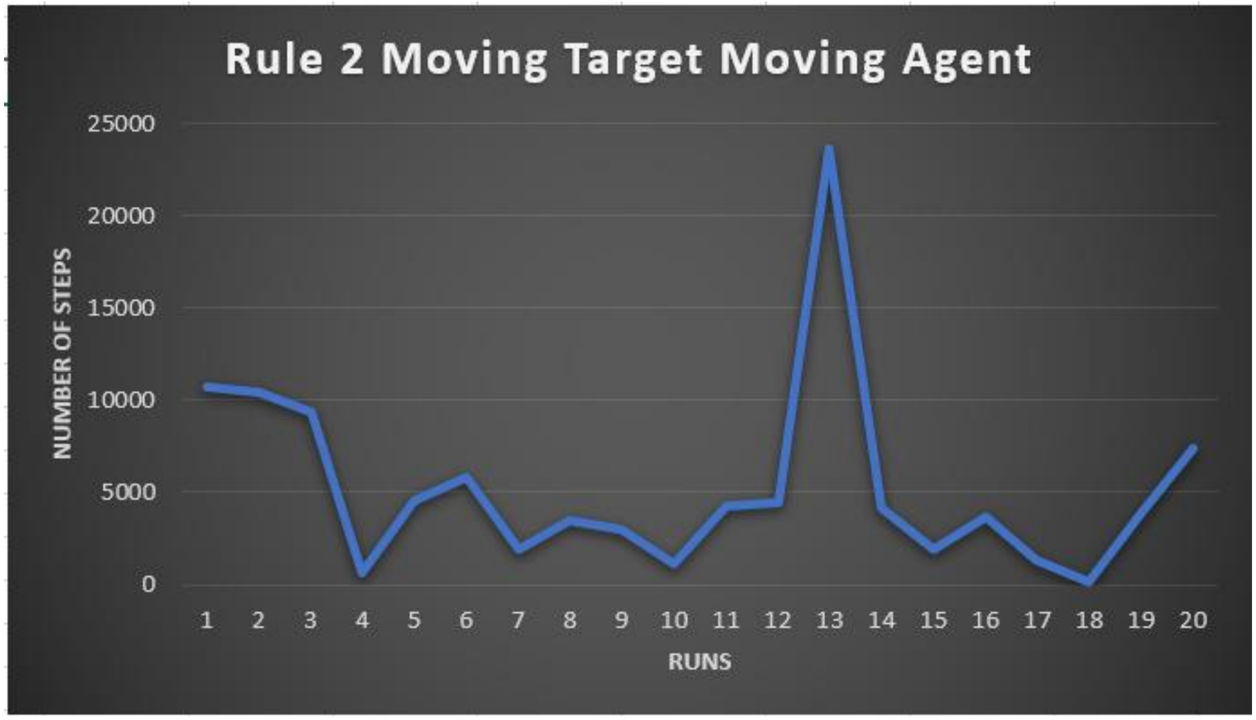
We also executed the Moving Target with the considering the Manhattan distance measure and got the following results for Rule1 and Rule2.

Rule 1:



Average number of search moves – 4062.4

Rule2:



Average number of search moves – 3948.9

In this case the Rule2 is slightly better than Rule1.