Kshitij Shah - ks1223
Sai Susmitha Batchu – sb1543

CS 520: INTRODUCTION TO ARTIFICIAL INTELLIGENCE

Assignment 1

Heuristic Search using Information from Many Heuristics

a) CREATING AN INTERFACE

We've created an interface using JavaFX that enables us to create, read, visualize and solve grid maps of dimension 160 columns and 120 rows. Every map contains different terrain types, namely, blocked cells, regular unblocked cells, hard to traverse cells, and highways with unblocked and hard to traverse cells. First, we initialize all the cells in the map to regular unblocked cells, then we randomly select eight coordinates and consider the 31x31 region centered around these coordinates in which every cell is hard to traverse with the probability 50%. In the next step, we select four non-intersecting paths that allow the agent to move faster along them (i.e., the "rivers" or highways). Then, we randomly select 20% of the total number of cells (which are not on highways) to mark as blocked. Finally, we randomly select the start and goal nodes. Maps will be stored in text files which contain the following information:

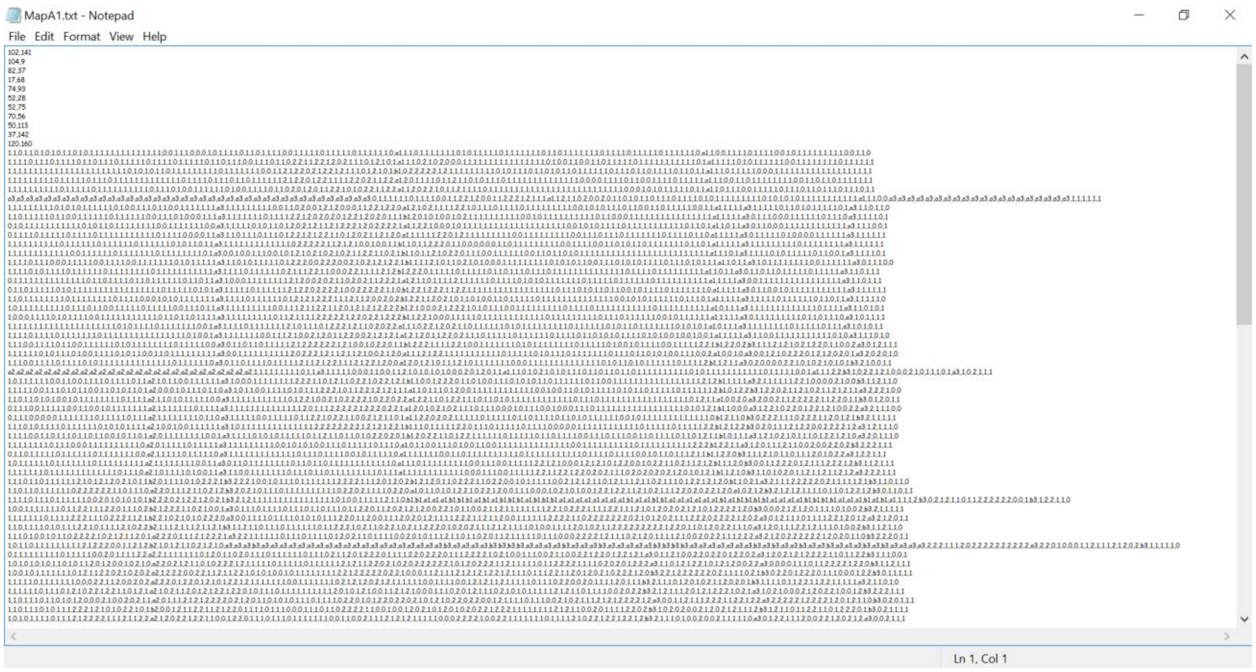
- The first line provides the coordinates of start node.
- The second line provides the coordinates of goal node.
- The next eight lines provide the coordinates of the centers of the hard to traverse regions.
- Then, we have 120 rows with 160 characters each that indicate the type of terrain:
  - ’0’ indicates a blocked cell
  - ’1’ indicates a regular unblocked cell
  - ’2’ indicates a hard to traverse cell
  - ’a_x’ indicates a regular unblocked cell on highway x
  - ’b_x’ indicates a hard to traverse cell on highway x

Figure 1 shows a text file containing a sample map.

We created five different maps with ten different start-goal pairs in each. After creating the maps, we run family of A* algorithms (uniform-cost, A* and weighted A*) to compute a path from the start node to goal node. These maps are visualized by using a color coding system to represent the various terrains:

- The start cell is represented by an aqua square.
- The goal cell is represented by a green square.
- Regular cells are represented by white squares.
- Blocked cells are represented by black squares.
- Hard to traverse cells are represented by grey squares.
- Regular cells on highway are represented by blue squares.

Kshitij Shah - ks1223
Sai Susmitha Batchu – sb1543

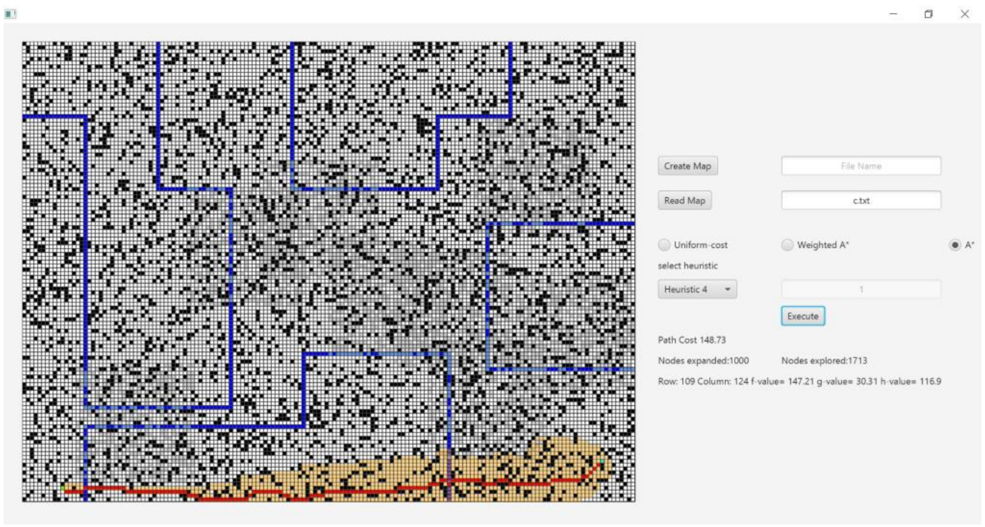


```
102,141
104,9
82,37
17,68
74,93
52,28
52,75
70,56
50,113
37,142
120,160
1,1,0,1,1,0,1,0,1,0,1,1,0,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,0,0,0,1,0,1,1,1,1,0,1,1,0,1,1,1,1,0,0,1,1,1,1,1,0,1,1,1,1,1,0,1,1,1,1,1,1,1,0,a1,1,1,0,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,0,1,1,1,[illegible]
1,1,1,1,0,1,1,1,0,1,1,1,1,0,1,1,0,1,1,1,0,1,1,1,1,1,0,1,1,1,1,0,1,1,1,1,0,1,1,0,1,1,1,0,0,1,1,1,0,1,1,0,2,2,1,1,2,2,1,2,0,2,1,1,1,0,1,2,1,0,1,a1,1,1,0,2,1,0,2,0,0,0,1,1,1,1,1,1,1,1,1,1,1,[illegible]
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,0,1,0,1,1,0,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,0,0,1,1,2,1,2,2,0,2,1,2,2,1,2,1,1,1,0,1,2,1,0,1,b1,0,2,2,2,2,2,2,1,2,1,1,1,1,1,1,1,1,1,1,0,1,0,1,1,[illegible]
1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,0,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,0,1,1,1,0,1,1,0,1,1,1,1,1,1,1,2,1,2,2,0,1,2,2,1,1,1,1,2,2,0,2,1,1,2,2,a1,2,0,1,1,1,1,0,1,1,2,1,1,0,1,0,1,1,1,0,1,1,1,[illegible]
1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,0,1,0,0,1,1,1,1,1,1,1,0,1,0,0,1,1,1,1,0,1,1,0,2,0,1,2,0,1,1,2,2,1,0,1,0,2,2,1,1,2,2,a1,1,2,0,2,2,1,0,1,1,2,1,1,1,1,0,1,1,1,1,1,1,[illegible]
a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,a3,0,1,1,1,1,1,1,0,1,1,1,1,0,0,1,1,2,2,1,2,0,0,1,1,2,2,2,[illegible]
[illegible]
```

Figure 1 File containing a Map

- Hard to traverse cells on highway are represented by light blue squares.
- The path is represented by red squares.
- The explored cells are tinted with light yellow.

Figure 2 shows the interface with the visualization of a sample map on which we have executed A*. This interface has options to create, read or execute a loaded map by selecting an algorithm and the parameters (such as heuristic and weight) it requires. We can a click on a particular cell on the visualized map to get its h, g and f values computed the A* family algorithms.



c.txt

( ) Uniform-cost ( ) Weighted A* [✓ A*]

select heuristic: Heuristic 4

1

Path Cost 148.73

Nodes expanded:1000 Nodes explored:1713

Row: 109 Column: 124 f-value= 147.21 g-value= 30.31 h-value= 116.9

Figure 2 Interface visualizing the map

b) ABSTRACT HEURISTIC ALGORITHM

The family of A* algorithms namely, Uniform-cost search, A* and Weighted A* have been implemented by using a generalized heuristic algorithm written in Java. The evaluation function we used is as follows:

$$f(s') = g(s') + w * h(s')$$

where, $g(s') = g(s) + c(s, s')$,

w is the weight

and $h(s')$ is the heuristic of node s'

The function g(s) represents the cost to reach the node s from the start node and $c(s, s')$ represents the cost to reach s' from s.

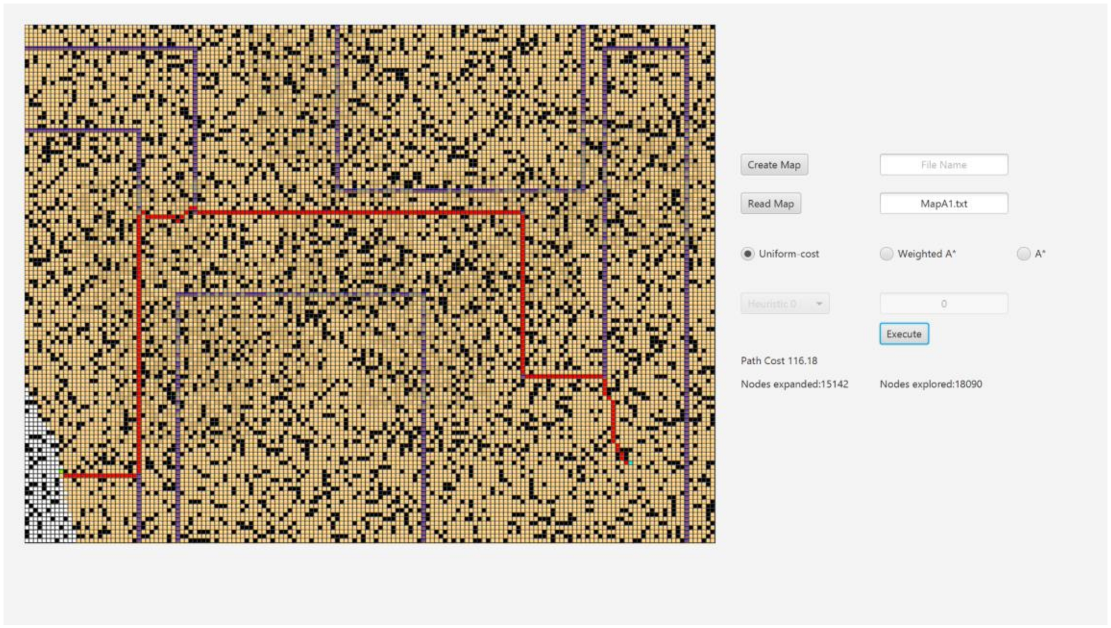
When we want to compute the path using Uniform-cost search, we set the heuristic to zero.

For executing A* on the map, as shown in Figure 2 from the interface, we can select the heuristic we want to use and the weight will be set to 1.

And for weighted A*, we can select both, the heuristic and its weight.

After execution, the generated path and the explored nodes will be shown on the map. The path cost, the number of nodes explored and the number of nodes expanded will also be displayed on the interface.

The following three figures show the execution of the three algorithms on the same map with same start and goal node (we use the same heuristic for A* and weighted A*).



MapA1.txt

[✓ Uniform-cost]

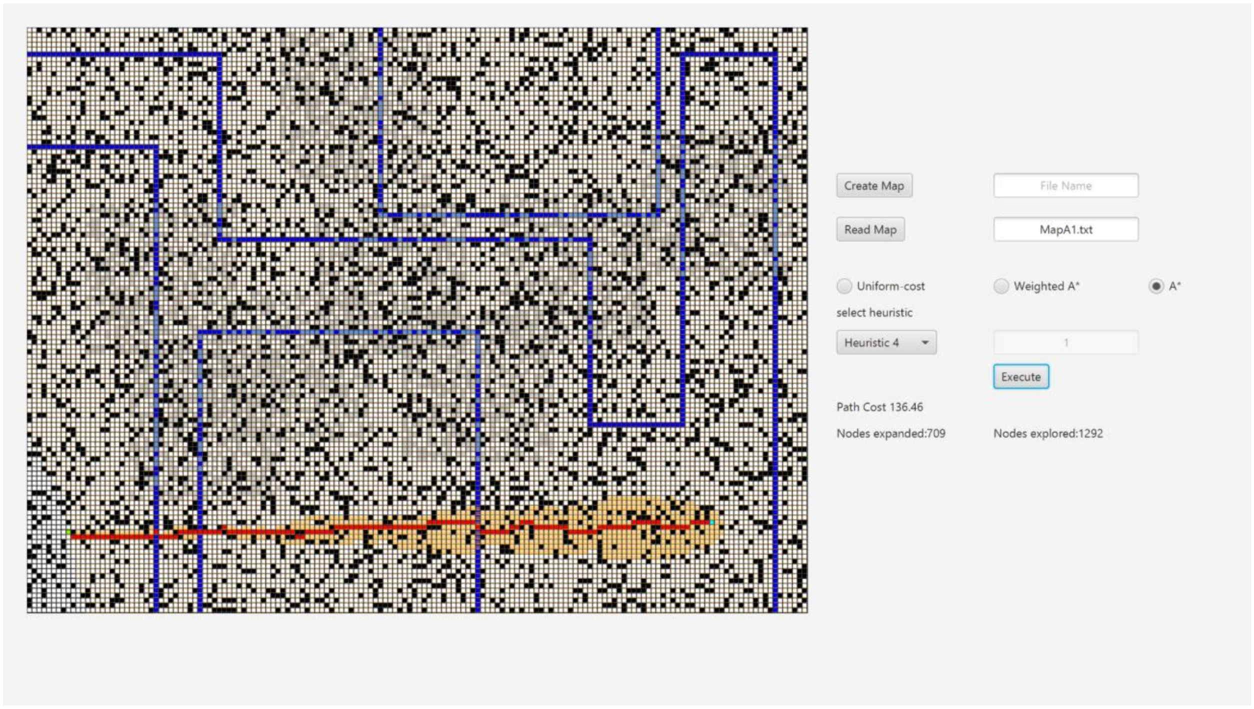
0

Path Cost 116.18

Nodes expanded:15142

Nodes explored:18090

Figure 3 Path using Uniform-cost Search

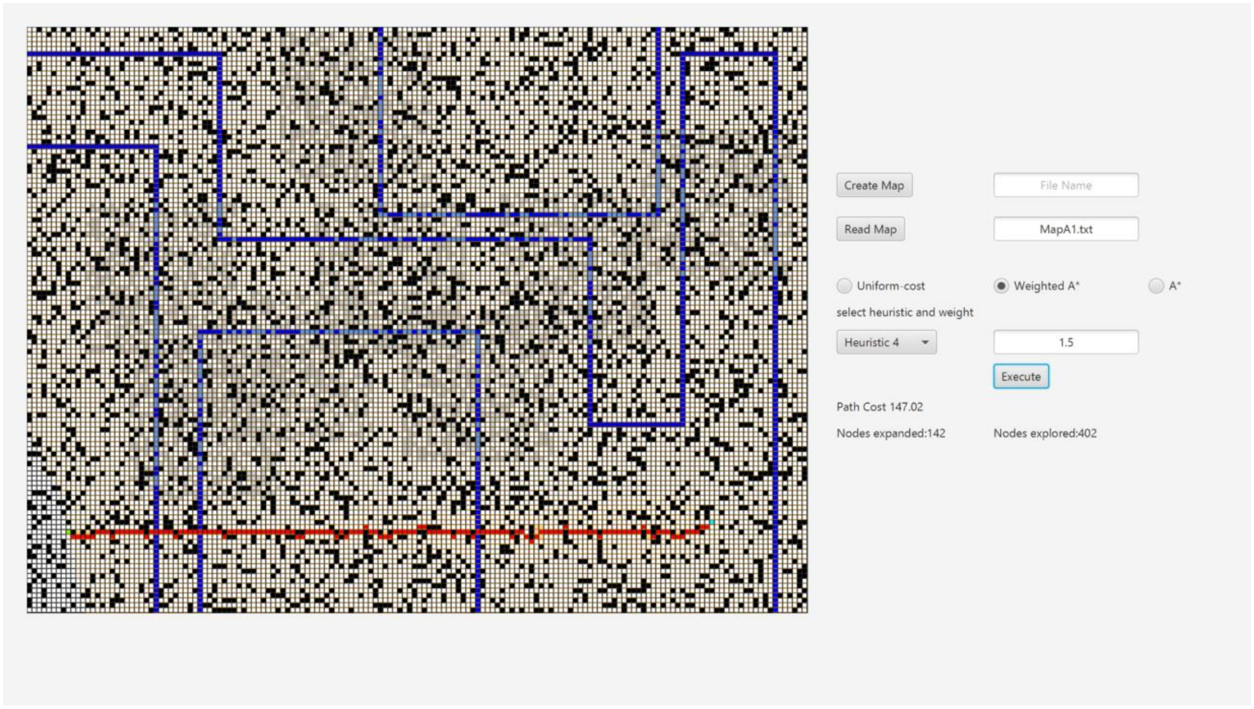


MapA1.txt

Weighted A* [✓ A*]
select heuristic
Heuristic 4
1

Path Cost 136.46
Nodes expanded:709
Nodes explored:1292

Figure 4 Path using A*



MapA1.txt
Uniform-cost [✓ Weighted A*] A*
select heuristic and weight
Heuristic 4
1.5

Path Cost 147.02
Nodes expanded:142
Nodes explored:402

Figure 5 Path using Weighted A*

c) OPTIMIZATION

Binary Heap

We implemented binary heap for fringe nodes in order to get the node with lowest f value in logarithmic time. There for cost of getting the node out of the fringe was significantly reduced, from n to $\log n$ to be precise, where n is the size of the fringe. However, the updating a specific node in the fringe still takes polynomial time with binary heap. On the practical note we found the running time with built-in heap structure of Java slightly better than our implementation. We suspect it uses more advance heap structure than binary heap.

Our binary heap used array of fixed size to store elements. It turns out that if numbered layer by layer starting from 1, children of the node k will be $2k$ and $2k + 1$. We used this property to access the children and parent of nodes, while a node is added or removed.

Closed List

In A* algorithm one has to check weather a node is in closed list or not. Nodes in the closed list are expanded and need not to be expanded again. However, checking for a specific node in closed list of size n requires n operations. As the size of the closed list grows it can lead to a significant overhead as this process need to be done for every generated successor. Instead we used a flag associated with every node which indicates weather the node is in closed list or not. Hence the time for checking gets reduced to constant time.

Ignoring Blocked Cells

We use a special condition to for not to add blocked cells in the fringe. Since, they are blocked no path can pass through blocked cells. Hence the size of fringe gets reduced and the cost for updating node in fringe is reduced.

d) HEURISTICS

We used two admissible and four inadmissible heuristics which are described below:

Due to the highways we cannot use straight line distances as an admissible heuristic. We used Manhattan distance on highways. As highways does not run diagonally it remains admissible.

Admissible

The best admissible heuristic we can find was cost to the nearest highway added to Manhattan distance from that highway to the goal. But finding nearest highway is not trivial problem, it requires exploring on its own which is undesirable for any heuristic function. A simplification of this heuristic is used instead. For any highway cell h is Manhattan distance on highway.

$$h = \ Manhattan\ distance\ on\ highway$$

$$h = Manhattan\ distnce\ from\ goal * 0.25$$

For other cell it's at least one added to the Manhattan distance from the Manhattan distance on highway from next cell. Resulting in,

$$h = 0.75 + Manhattan\ distance\ on\ highway$$

Inadmissible

For inadmissible heuristic we use straight line distance as Manhattan distance base. Another heuristic we used uses the path length on plain grid as heuristic. That is diagonal movement as long as possible followed by straight movement. We used combination of this distances to get better results. We used the following formulas to calculate distances.

$$Manhattan\ distance = (x_{goal} - x_{node}) + (y_{goal} - y_{node})$$

$$Straight\ line\ distance = \sqrt{(x_{goal} - x_{node})^2 + (y_{goal} - y_{node})^2}$$

$$\begin{aligned} Diagonal\ distance \\ &= \min\big((x_{goal} - x_{node}\big), (y_{goal} - y_{node})) \\ &+ absolute\ difference\big((x_{goal} - x_{node}\big), (y_{goal} - y_{node})) \end{aligned}$$

We also worked on using the current cells status in order to get better approximation of path cost. However, we have not used surrounding information in our program.

f) RESULTS AND OBSERVATIONS

Uniform-cost and A*will return the same path cost when we use an admissible heuristic for A*. But, the drawback with Uniform-cost is that it explores almost all the nodes of the map and hence will have more running time compared to the other two. This can be a drawback for real time applications where the maps may sometimes have several thousand grids.

Weighted A* will explore even lesser nodes than A* but it will return a suboptimal path which costs more than the optimal path returned by the other two algorithms. The cost increases in proportion the weight of the heuristic.