Programming Assignment 2

Heuristic based Problem Solving Agent

Problem Description : Consider the following minesweeper problem. A rectangular grid of squares is a war field with a number of mines which may explode if clicked [Fig.1]. The mines are not visible to the intelligent agent walking over the area. The agent has a radar based device which detects the number of mines in the vicinity of the square. A square has maximum 8 neighbors in its horizontal, vertical and diagonal directions which may contain one or more mines.

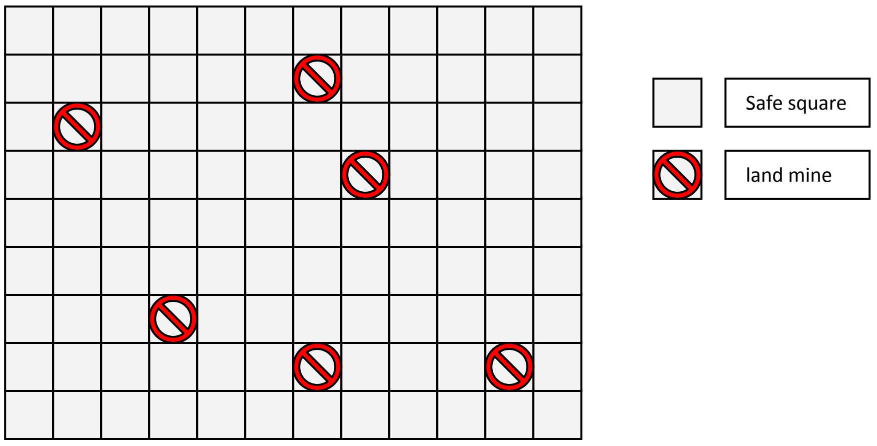

Figure1: Mines and safe squares

The agent is said to be in a state represented by a n-valued tuple as discussed in the class. The mines are not dynamic and remain at the same place throughout the search in a run. Once clicked in a square that does not have any mine, the neighboring safe area opens up with the boundary squares. The open area does not have any mine in it. The area that opens up as safe zone is a convex region bounded by the nearest mines in horizontal and vertical directions. The convex region is shown in Figure 2. The area is not convex if it has a bend as shown in Figure 3.

Each move refers to a click at any one remaining square outside the open area followed by further expansion of the open area. A wrong click at hidden mine may cause an explosion, therefore, the walking agent has to take intelligent decisions based on the number of mines obtained for squares at the border of the open area. Figure 4 shows the state of the field after first click.

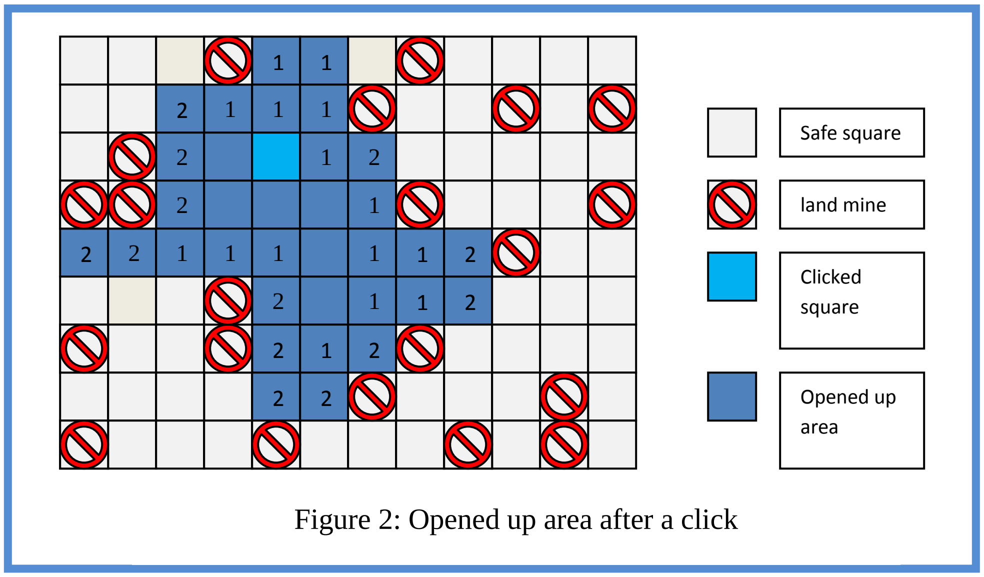

Figure 2: Opened up area after a click

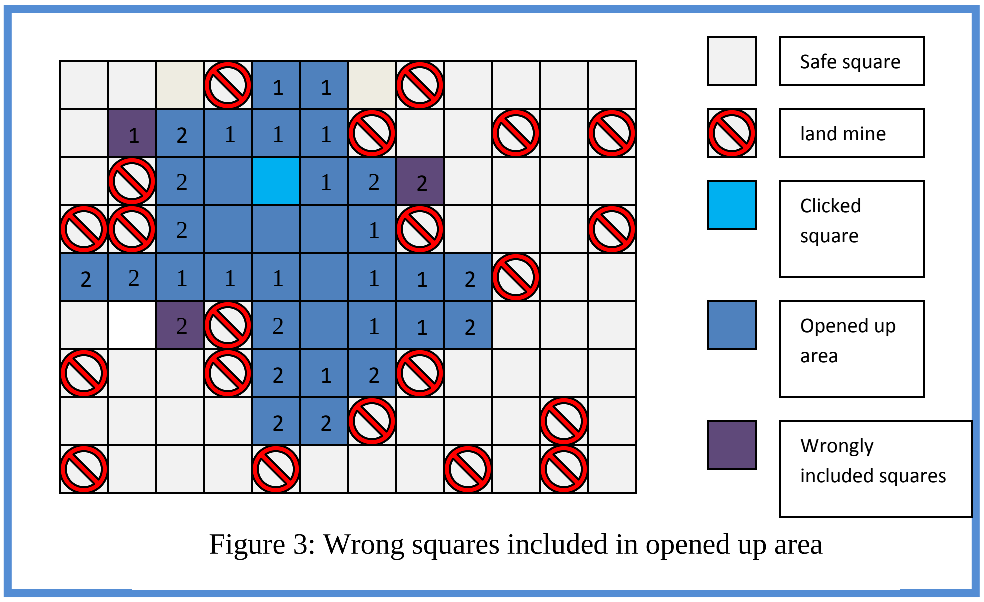

Figure 3: Wrong squares included in opened up area

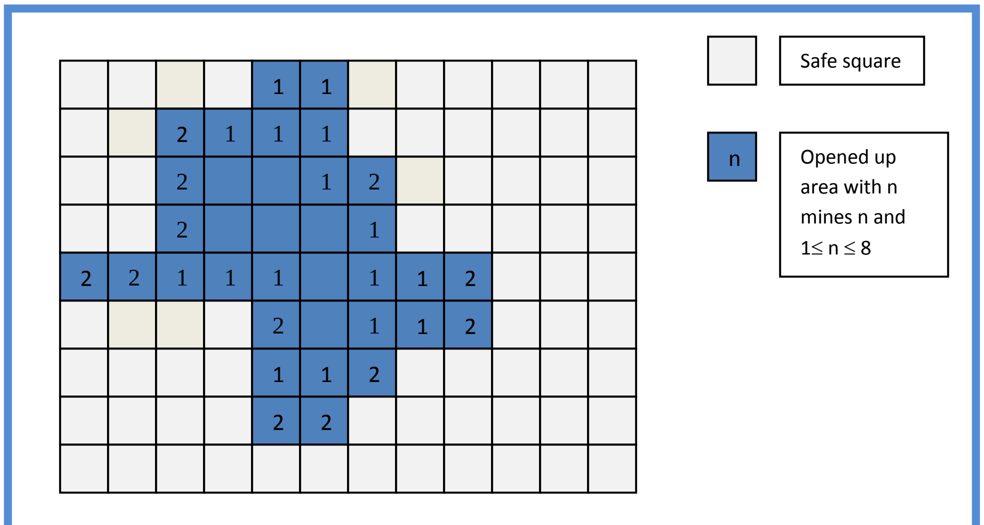

Figure 4: State of the field after the first click

The state of the field after second move is shown in Figure 5

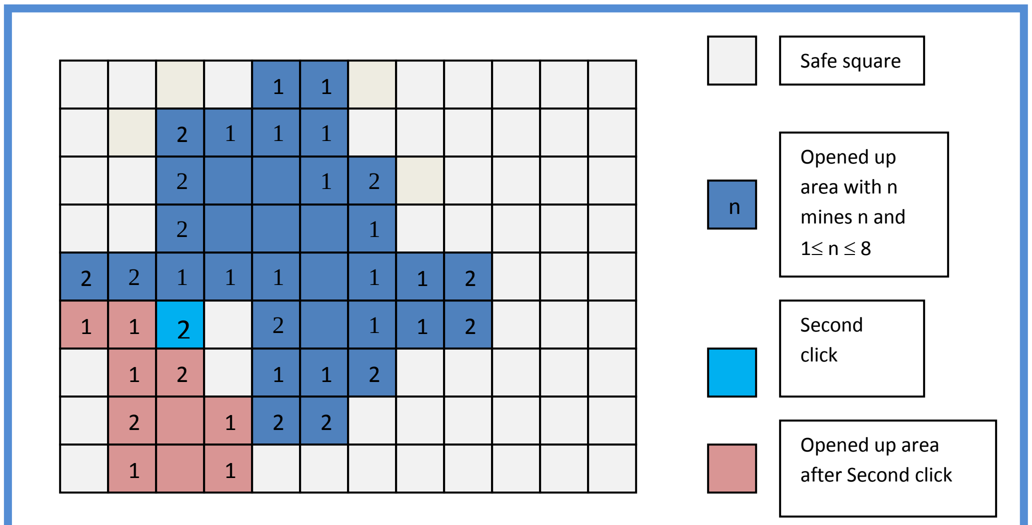
Safe square
n
Opened up area with n mines n and 1≤ n ≤ 8
Second click
Opened up area after Second click

Figure 5: State of the field after the second click

Similarly further moves are made based on the heuristic appropriately. Represent the above problem as a state space search problem. Define appropriate heuristic for selecting the next move. Also represent the problem as optimization problem and observe the nature of optimization problem: maximization or minimization. Also, observe the heuristic value of the goal, how far it is from the initial state, any local optima trap you are seeing and so on. Observe the difference between the performances of the two heuristics by working out the problem on paper as well to initial few steps.

Understand how beautifully the heuristic values represent human intelligence (use your own and see if you would do the same in making a move).

Implementation

Use Python version 3.7 (Windows 10), PyQT and matplotlob for implementing your solution. Only standard Python libraries should be used. Support from external libraries such as github will not be accepted in your submissions. Each student must design own solution and write own code. [Refer handout to understand the malpractice policies.]

Modules

You must implement the following in your program.

1. Function **mineGenerator (size_warfield N, unsigned int M):** The function generates a random arrangement of M mines and returns a state of the war field of size N (specified number of rows and columns) with hidden mines. Mines once generated remain part of the state throughout in one run of the algorithm.
2. Function **firstClick(state S):** The function takes the first step by clicking blindly on any square. If the clicked square is a mine, then the mine explodes. But if there is no mine, then the function produces the next state with opened up area and the number of mines appropriately.
3. Function **nextState(state s, action A)**: Computes the next state as discussed.
4. Functions **heuristicFunction_h1(state s)** and **heuristicFunction_h2(state s)**: Computes the heuristic value of the state and returns an integer value. You are required to define two significant heuristics say h1 and h2.
5. **Informed Search techniques**
   Implement the following techniques (say T1 and T2)
   - Hill Climbing Search (With variations such as stochastic HC, random restart and local beam search] [T1]
   - Simulated Annealing (with and without threshold)[T2]

6. **Graphics**
   Use Python based Turtle graphics, PyQT or matplotlib appropriately to display the current state with opened up area and showing the graphs etc. appropriately. First display the initial state of the problem. Keep showing the expanded open area graphically as you proceed with more computations with selected node and mark the goal when reached. Second, a user friendly comprehensive Graphics User Interface (GUI) is required to be created. Create a convenient main menu through which you can navigate the display and can revisit a display.

7. **Analysis Module**

Produce the following analyses and display the resultant values on the GUI.

(a) Hill Climbing (T1) based analysis

i. **Memory**: Compute the amount of maximum memory allocated till the problem is solved. **[R1]**

ii. **Time:** Compute the time to reach the goal from the randomly generated initial state **[R2].**

iii. **Solution:** For graphics, use an appropriate method to show the expansion of open area using the technique T1 for each move. Display the whole war field with opened up area in **green** color, unexplored squares in **grey** color and hidden mines in **red** colored icons as used in above description (with 30% transparency) and the selected click in **Blue** color. Ensure that the next move is shown once the viewer presses a key "Next move" after carefully seeing your previous move. Keep mentioning the status of the move [e.g. as 5th move]. Remember a move is from current state (say c) to the next state and should not be misunderstood as just a click. Mention the value of the heuristic for each move and display in self explanatory manner to justify the move. Notify on the screen, your heuristic computation for the selected move (say m) and that of all others (say n) whose difference of heuristic values [i.e. h(n) - h(c), if problem is posed as a maximization problem or as appropriate] is lesser than that of the selected move with the difference h(m)-h(c). Mention on the screen the branching factor for the problem.**[G1]**

iv. **Local Optimal solution:** Do not terminate if you accidently click on the mine. Count that as a sub optimal solution and show a termination of your algorithm at local optima. Showcase more information to justify the local optima with heuristic value of the state reached appropriately. **[R3]**

v. **Behavior of the Hill Climbing algorithm using h1:** Randomly create the mines 20 times (Random restart) and report the total number of times the algorithm reached the global optima using heuristic h1. **[R4]**

vi. **Behavior of the Hill Climbing algorithm using h2:** Randomly create the mines 20 times and report the total number of times the algorithm reached the global optima using heuristic h2. **[R5]**

vii. **Number of steps:** Report the average number of steps taken by the Hill Climbing algorithm in 20 independent runs till its termination at local or global optima. **[R6]**

viii. **Landscape:** Allow the viewer to select any two dimensions of the state and plot a 3D surface using matplotlib, where the third dimension is the heuristic value of the full state representation. Keep this in a loop till the viewer wishes to observe the landscape for any combination of the values (say x_2 and x_5 from a full state<x_1, x_2, x_3, x_4, x_5>). Give the viewer a choice for selecting the heuristic (h1 or h2 with appropriate explanation on the screen) **[G2]**

ix. **Behavior of the Stochastic Hill Climbing algorithm using h1:** Randomly create the mines 20 times (Random restart) and report the total number of times the algorithm reached the global optima using heuristic h1. **[R7]**

x. **Behavior of the Stochastic Hill Climbing algorithm using h2:** Randomly create the mines 20 times and report the total number of times the algorithm reached the global optima using heuristic h2. **[R8]**

xi. **Behavior of the local beam search algorithm using h1:** Randomly create the mines 20 (k=20) times and report the total number steps taken by the local beam search algorithm to reach the global/ local optima using heuristic h1. Also mention whether reached local or global optima. **[R9]**

xii. **Behavior of the local beam search algorithm using h2:** Randomly create the mines 8 (k=8) times and report the total number steps taken by the local beam search algorithm to reach the global/ local optima (convergence) using heuristic h2. Also mention whether reached local or global optima. **[R10]**

xiii. **Effect of varying value of k in local beam search:** Use matplotlib to plot the number of steps taken by local beam search (on Y axis) by varying value of k from 1 to 100 (on X axis). Plot using Red color for h1 heuristic and use Blue color for h2 heuristic. **[G3]**

xiv. **Effect of size of war field:** Make changes in the size of war field from 10x10 to 100 x 100 taking equal steps of 10 in both number of rows and number of columns. Study the effect by plotting the average number of times the algorithm reached the global optima in 10 independent runs. **[G4]**

(b) Simulated Annealing (T2) based analysis

i. **Memory**: Compute the amount of maximum memory allocated till the problem is solved. **[R11]**

ii. **Time:** Compute the time to reach the goal from the randomly generated initial state **[R12].**

iii. **Solution:** For graphics, use an appropriate method to show the expansion of open area using the technique T1 for each move. Display the whole war field with opened up area in **green** color, unexplored squares in **grey** color and hidden mines in **red** colored icons as used in above description (with 30% transparency) and the selected click in **Blue** color. Ensure that the next move is shown once the viewer presses a key "Next move" after carefully seeing your previous move. Keep mentioning the status of the move [e.g. as 5th move] and mention if this is a **bad** move. Remember a move is from current state of the war field (say c) to the next state and should not be misunderstood as just a click alone. Mention the value of the heuristic for each move and display in self explanatory manner to justify the move. Mention on the screen the branching factor for the problem.**[G5]**

iv. **Local Optimal solution:** Do not terminate if you accidently click on the mine. Count that as a sub optimal solution and show a termination of your algorithm at local optima. Showcase more information to justify the local optima with heuristic value of the state reached appropriately. **[R13]**

v. **Behavior of the Simulated annealing algorithm using h1:** Randomly create the mines 20 times (Random restart) and report the total number of times the algorithm reached the global optima using heuristic h1. **[R14]**

vi. **Behavior of the Simulated annealing algorithm using h2:** Randomly create the mines 20 times and report the total number of times the algorithm reached the global optima using heuristic h2. **[R15]**

vii. **Number of steps:** Report the average number of steps taken by the Hill Climbing algorithm in 20 independent runs till its termination at local or global optima. **[R16]**

viii. **Landscape:** Allow the viewer to select any two dimensions of the state and plot a 3D surface using matplotlib, where the third dimension is the heuristic value of the full state representation. Keep this in a loop till the viewer wishes to observe the landscape for any combination of the values (say x_2 and x_5 from a full state<x_1, x_2, x_3, x_4, x_5>). Give the viewer a choice for selecting the heuristic (h1 or h2 with appropriate explanation on the screen) **[G6]**

ix. **Temperature range:** Drape the movement of the selected states on the 2 dimensional landscape (od selected dimensions) with decreasing temperatures as a sequence of lines from start to end especially to showcase the bad moves and ability to reach global solutions. **[G7]**

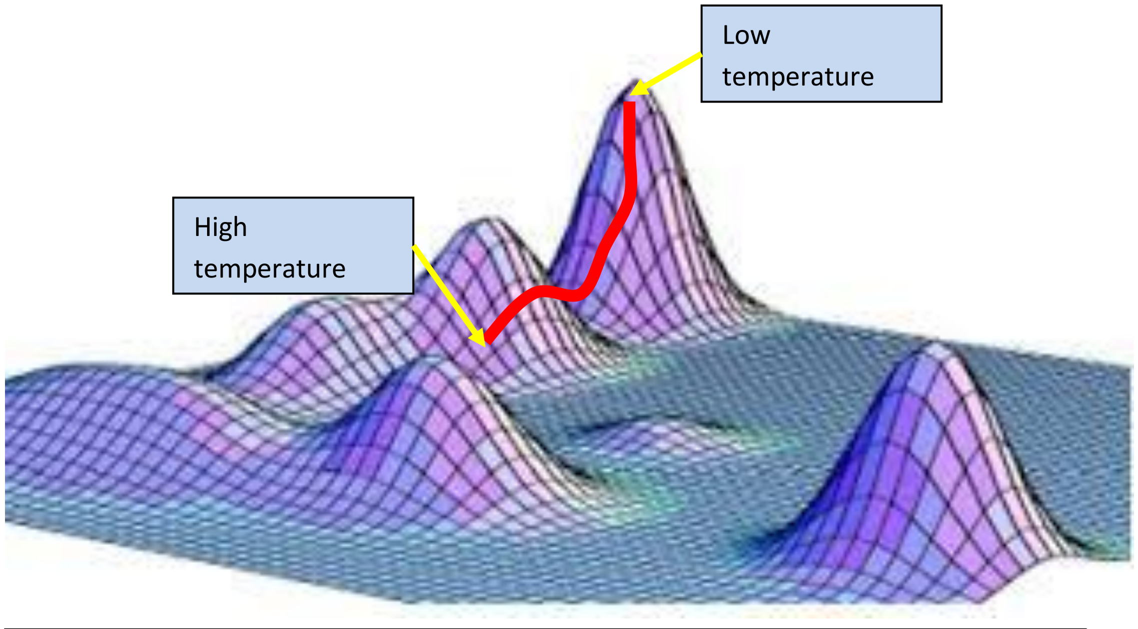

Figure 6: Draping of the movements on a 3D landscape as the temperature decreases

8. Driver

The driver must integrate all functionalities and execute the functions appropriately using interactive graphics display as described above. Use selection buttons for heuristics, receive input about the size of the war field, number of mines in the war field etc from the user. Show all pre computed values R1 to R16 and show the graphs G1 to G7 as asked. Also provide ways to receive input for executing your code at my end as well.

Writeup, evaluation and submission

Write up details will be made available a day before the submission. Evaluation will be out of 16 marks (8% weight). Students are advised to inform me immediately if any discrepancy exists in this document. The assignment is due for submission **on September 27, 2019 (Friday) by 6:00 p.m**.

All students are advised to work out the details of the solution and plan its implementation appropriately. Timely start, sound theoretical understanding of the subject, regularity in attending lectures, sincerity in applying the learned knowledge in problem solving, doing home work in time and an interest in the subject are the key parameters which contribute in the comfortable and smooth completion of the work given as assignment.

Please inform me any discrepancy in the above text at the earliest. In case of any difficulty, please feel free to contact me.

Vandana

September 16, 2019
