Minesweeper

Contributor:

Malhar Khimsaria (mk200)

Kushal Doshi (kd663)

Vivek Dhandha (vd264)

Ajinkya Patankar (aap256)

3) Questions and Writeup

Representation:

The Minesweeper board is a 2D array with the user defined dimensions. The individual cells are identified by one of the either Boolean variables, whereby 1 represents a mine and 0 represents no mine.

For instance:

		a	Z(1)
		b	c

In the above board, Z(1) represents the starting cell. This can be selected at random and does not require any predefined measure. The (2) represents that out of the neighbours, a, b, c only 2 can represent mines while third is free cell.

These 2 mines can be anywhere in 3 cells, which is calculated by ${}^{3}C_{2}$ combinations. These are:

a	b	c	Z(2)
1	0	0	0
0	0	1	0
0	1	0	0

Now, after traversing the start cell, we can make either of the three choices with us. Suppose, we select b, such that it does not have a mine, then, we must also find similar combinations for b's neighbours.

	d	a	Z(1)
	e	B(2)	c
	F	g	h

Since, from previous selection, we realise that either a or c will definitely be mines, given b is not, we can safely assume that our choices of combinations lie amongst d, e, f, g, h, a, c. Now, we draw up combinations amongst the remaining neighbours.

d	e	f	g	h	a	C	B
1	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0
1	0	0	1	0	0	0	0
1	0	0	0	1	0	0	0
1	0	0	0	0	1	0	0
1	0	0	0	0	0	1	0
0	1	1	0	0	0	0	0
0	1	0	1	0	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	0	1	0	0
0	1	0	0	0	0	1	0

0	0	1	1	0	0	0	0
0	0	1	0	1	0	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	0	1	0
0	0	0	1	1	0	0	0
0	0	0	1	0	1	0	0
0	0	0	1	0	0	1	0
0	0	0	0	1	1	0	0
0	0	0	0	1	0	1	0
0	0	0	0	0	1	1	0

Now, we have to extract combinations of the two tables: instances where b = 0, a = 1, c = 0 and b = 0, a = 0, c = 1.

d	E	f	g	h	a	C	B
1	0	0	0	0	0	1	0
1	0	0	0	0	1	0	0
1	0	0	0	0	0	1	0
0	1	0	0	0	1	0	0

0	1	0	0	0	0	1	0
0	0	1	0	0	1	0	0
0	0	1	0	0	0	1	0
0	0	0	1	0	1	0	0
0	0	0	1	0	0	1	0
0	0	0	0	1	1	0	0
0	0	0	0	1	0	1	0

Until here, we have taken into consideration every possible scenario that might occur. This terminates the possibility of an unseen case, the program code might encounter in the midst of execution.

Decisions:

Now, we have to calculate the probability of program code encountering a mine for each of the above possible scenarios.

$P(d) = (1+1+1+0+0+0+0+0+0+0+0)/11 = 3/11 = 0.27$

$P(e) = (0+0+0+1+1+0+0+0+0+0+0)/11 = 2/11 = 0.18$

$P(f) = (0+0+0+0+0+1+1+0+0+0+0)/11 = 2/11 = 0.18$

$P(g) = (0+0+0+0+0+0+0+1+1+0+0)/11 = 2/11 = 0.18$

$P(h) = (0+0+0+0+0+0+0+0+0+1+1)/11 = 2/11 = 0.18$

$P(a) = (0+1+0+1+0+1+0+1+0+1+0)/11 = 5/11 = 0.45$

$P(c) = (1+0+1+0+1+0+1+0+1+0+1)/11 = 6/11 = 0.55$

After finding out the probabilities, we decide which cell to visit next. This we do by considering the probabilities of the cells. Ideally, we would decide a prob threshold p0:

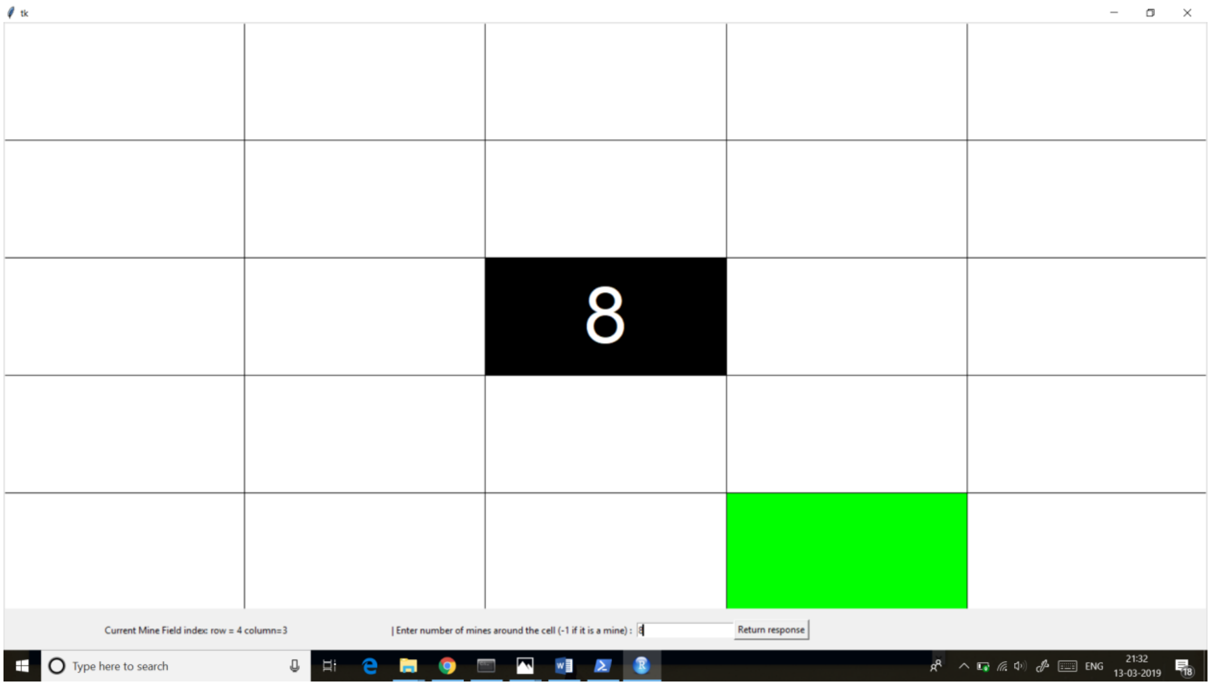
1) If $p <= p_0$, then we would select the cell with prob p. If there are multiple cells with same p, we move to the cell whose neighbours are most explored or known.
2) If $p > p_0$, and if p is sufficiently high, then it might be the case, that we cannot open that cell because the risk of opening that cell is too high.

There is no fixed way of determining p_0, since this is very problem specific and varies with the mine density. Instead, what we do is in order to select the next cell, we select the cell with min probability and maximum neighbours explored.

Performance

Mine to be solved

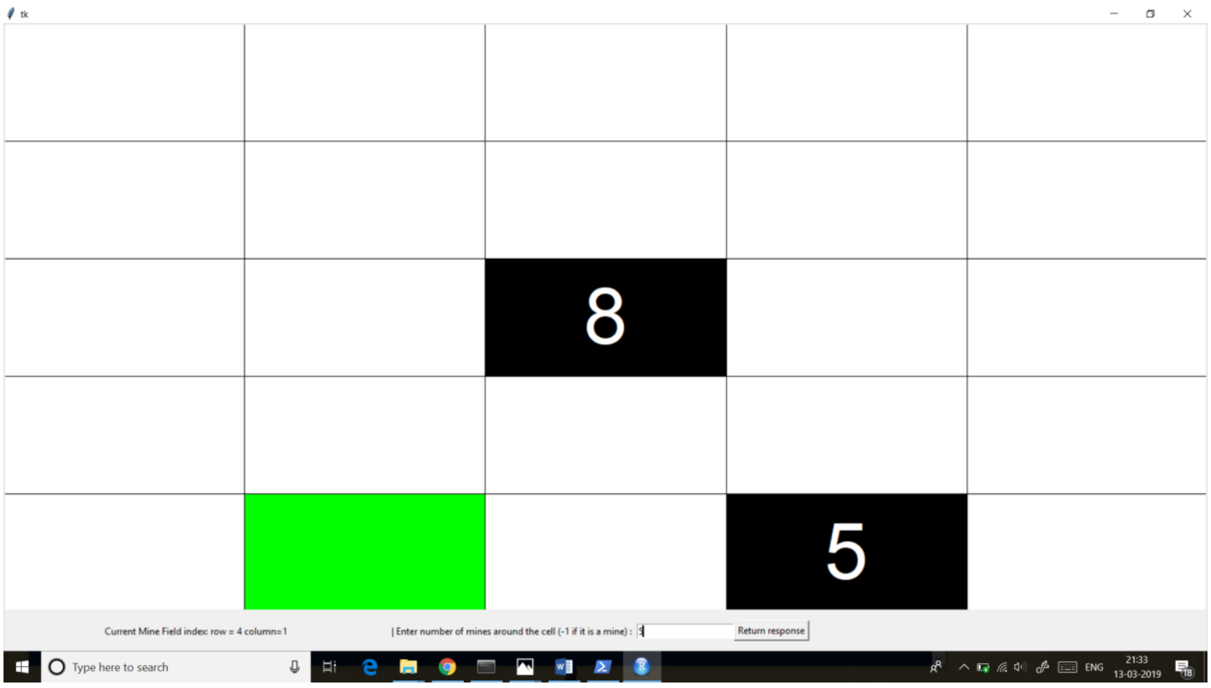
-1	5	-1	5	-1
-1	-1	-1	-1	-1
5	-1	8	-1	5
-1	-1	-1	-1	-1
-1	5	-1	5	-1



Current Mine Field index: row = 4 column=3

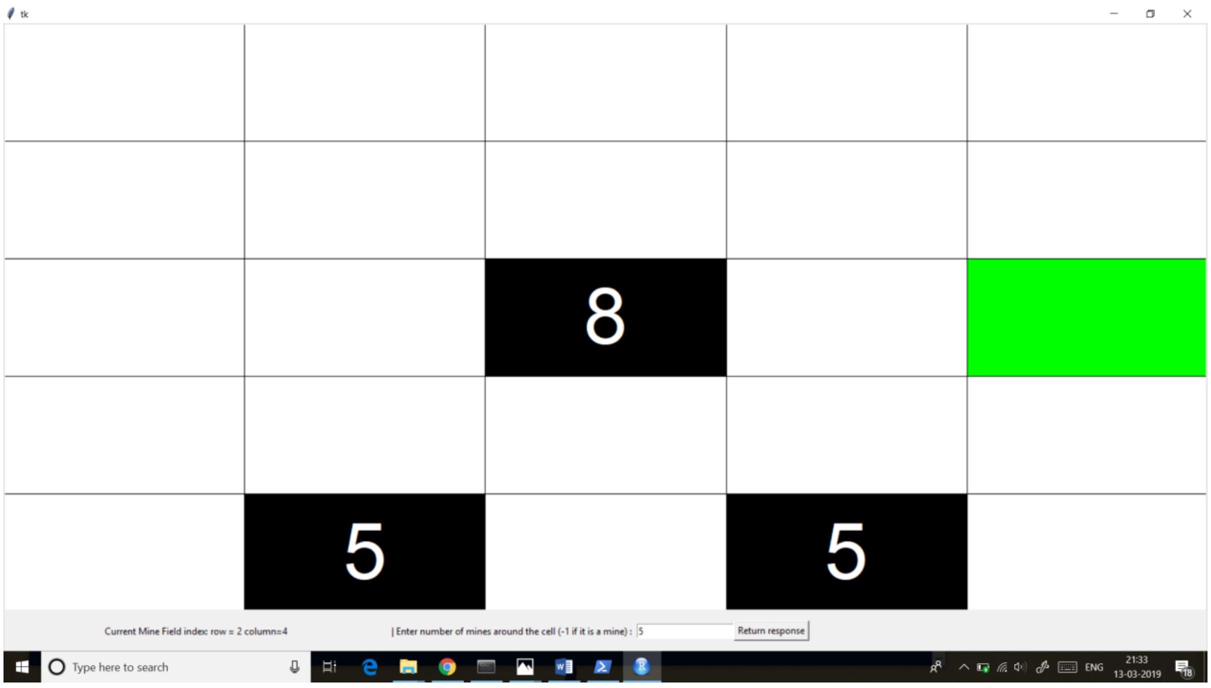
| Enter number of mines around the cell (-1 if it is a mine) : 8

Return response



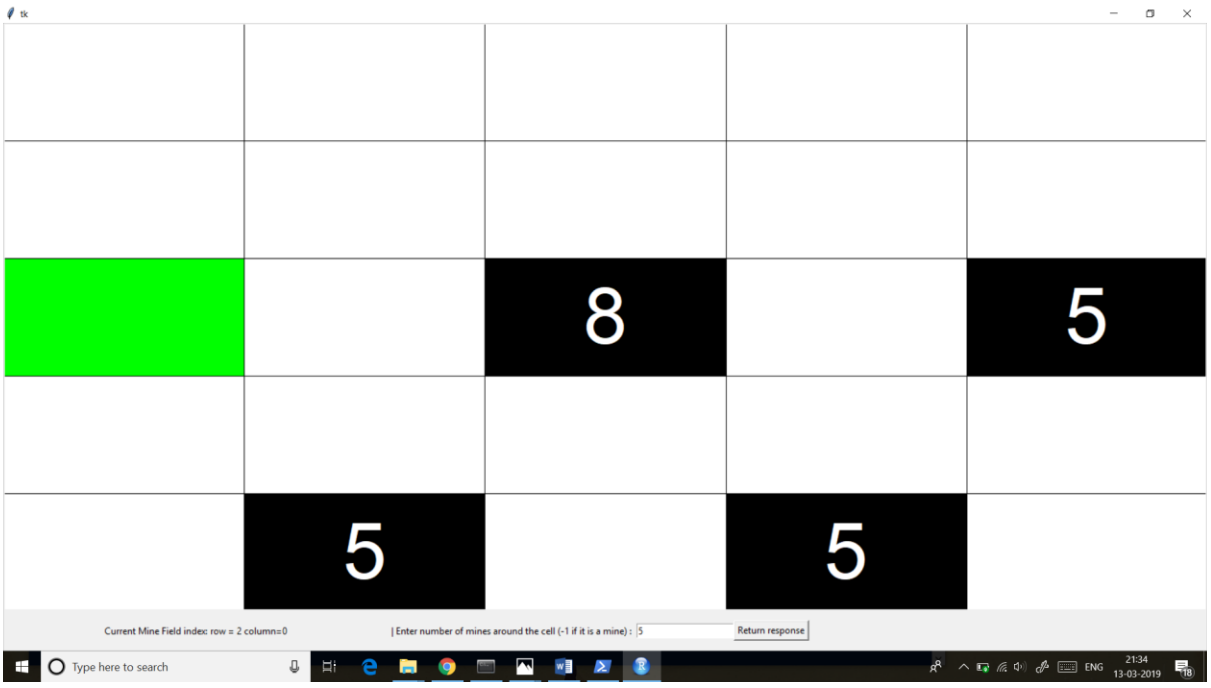
Current Mine Field index: row = 4 column=1

| Enter number of mines around the cell (-1 if it is a mine) : 5



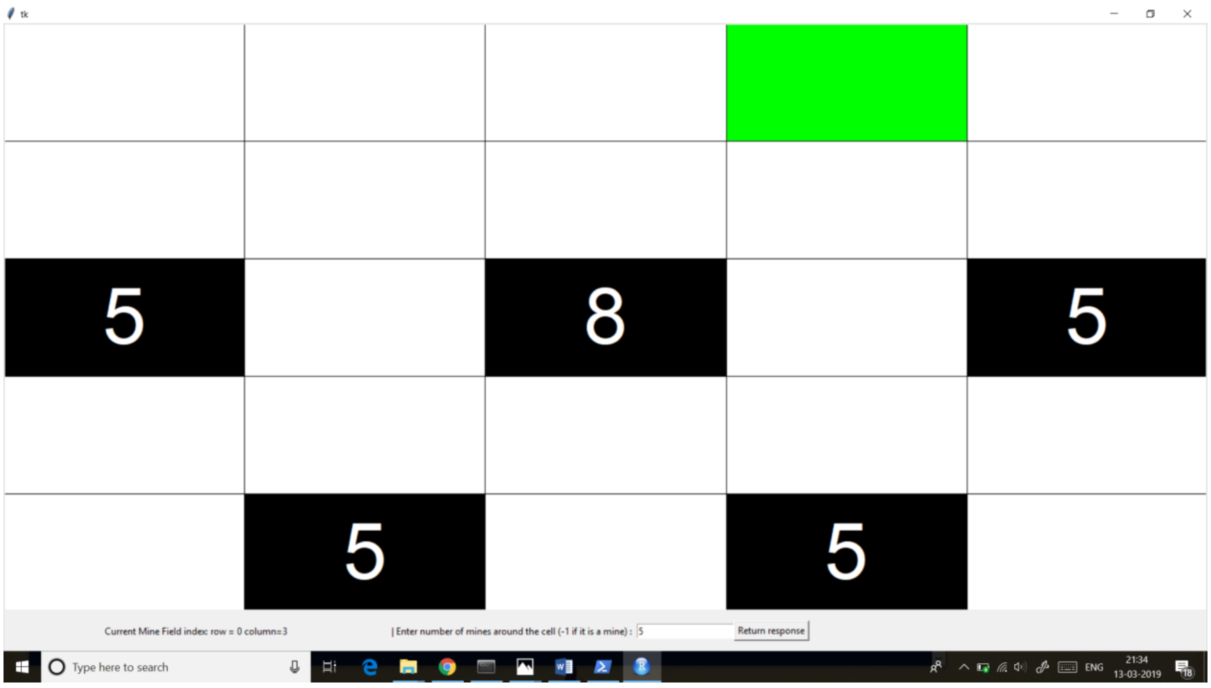
Current Mine Field index: row = 2 column=4

| Enter number of mines around the cell (-1 if it is a mine) : 5



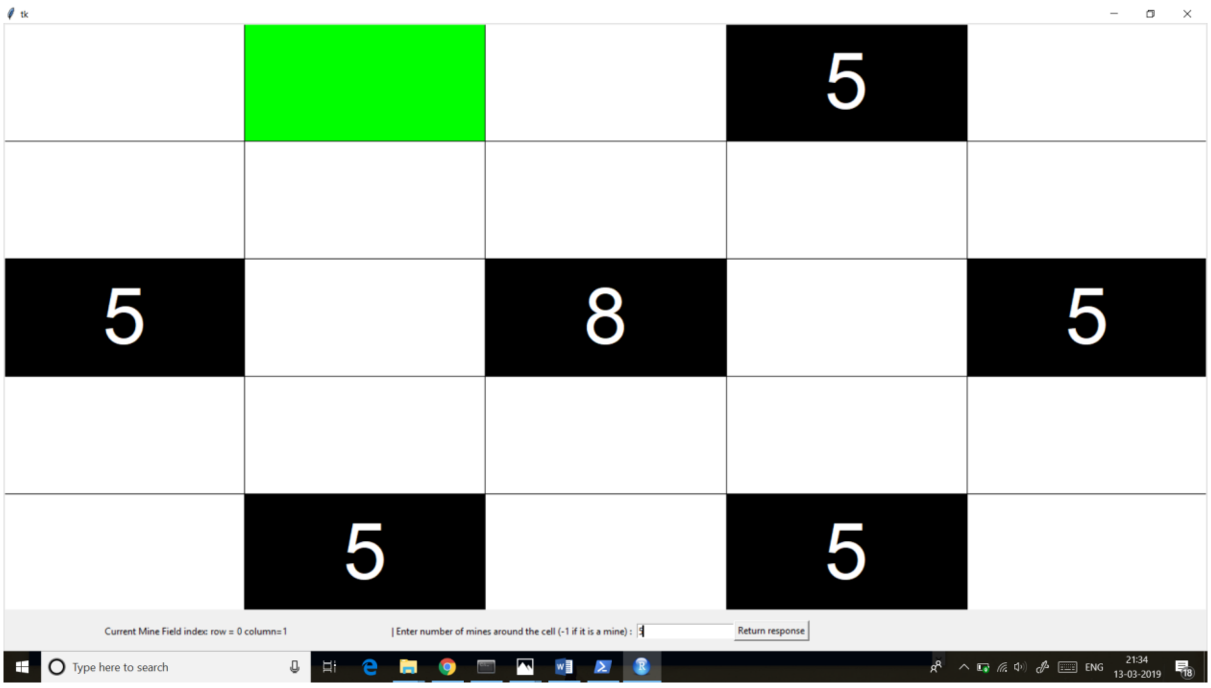
Current Mine Field index: row = 2 column=0

| Enter number of mines around the cell (-1 if it is a mine) : 5 Return response



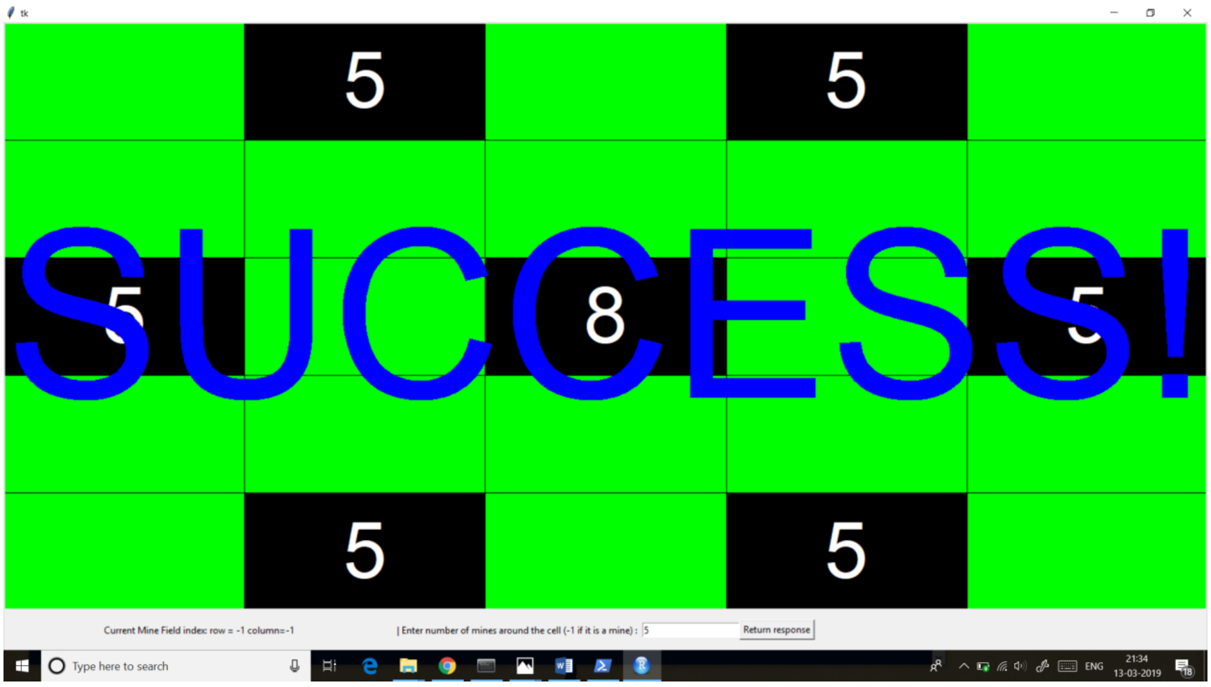
Current Mine Field index: row = 0 column=3

| Enter number of mines around the cell (-1 if it is a mine) : 5 Return response



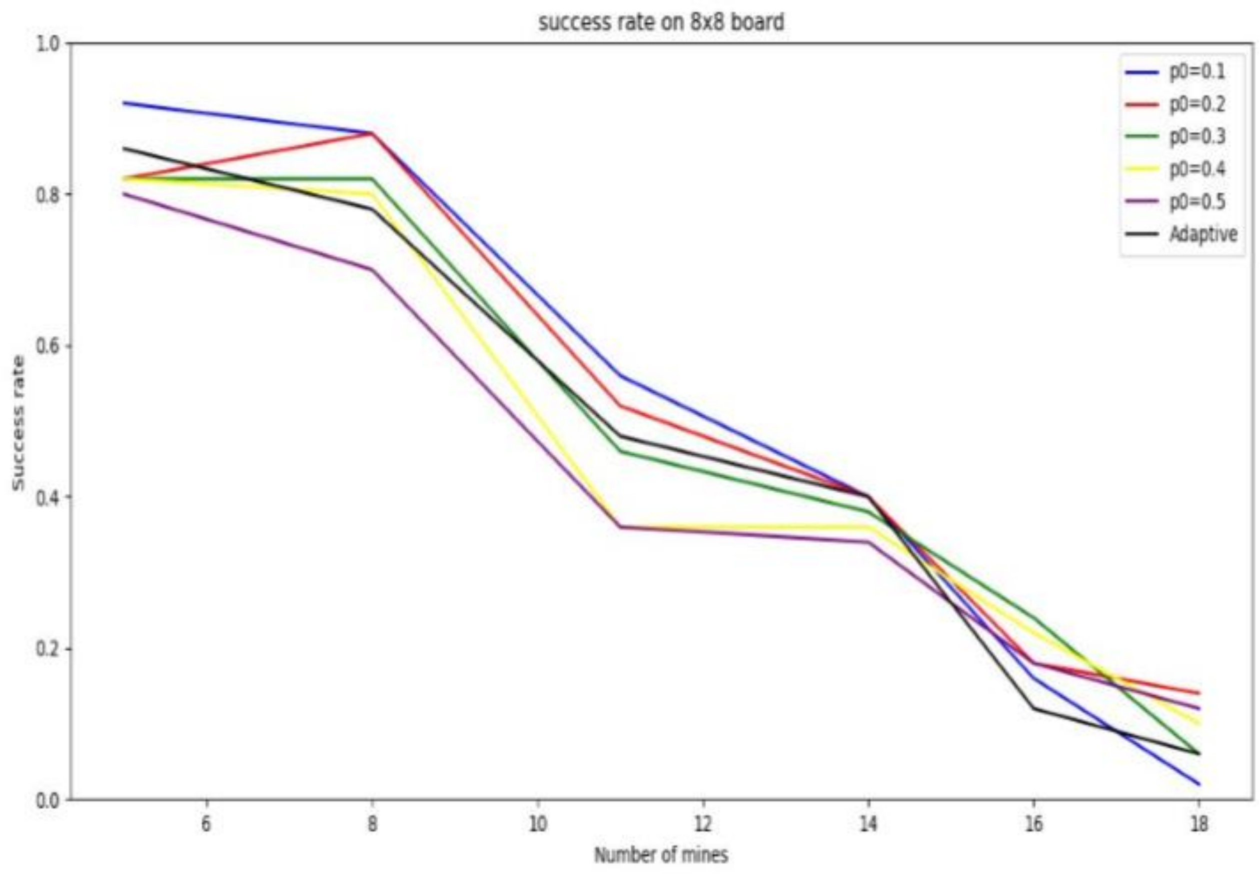
Current Mine Field index: row = 0 column=1

| Enter number of mines around the cell (-1 if it is a mine) : 5



Current Mine Field index: row = -1 column=-1

| Enter number of mines around the cell (-1 if it is a mine) : 5



We run benchmarking tests on a 8*8 board. The probability p0 is the threshold which we use to select the next cell. Whereas in the adaptive techniques, we select the next cell with the lowest prob and maximum number of visible neighbours. For each of the increasing number of mines, chances of success goes on decreasing. When the number of mines increases, after 14 or more mines the success rate goes down below 20%. This is because the algorithm makes a lot of random moves when the number of mines is high

Efficiency

For the large sized mazes, the combinations to identify the next moves is quite large, and the algorithm relies on guess work, as it is uncertain about the next move. This issue is quite problem specific since the mine field doesn't provide us enough information to make the next move.

We can work around this problem, by considering every cell with $p > 0.75$ as a mine and $p < 0.25$ as a clear cell. However, this strategy is untested as of now, and may prove to be more detrimental than productive.

Improvements

If we know the number of mines, we can be sure about when to terminate our search. We can create a global variable, so that we can track the number of mines. The counter starts with 0, and is incremented by 1, each time a mine is encountered. Once the value becomes reaches the exact number of mines, we can abandon the search and declare all the unexplored cells as safe.