MineSweeper 2.5 - Decision Making Report

CS 440

```
[2][A][1]
[B][C][D]
[E][3][F]
```

Cesar Herrera, Jae Weon Kim

Spring 2020

List of Figures

Assignment 2.5 - Decision Making

This project is inteded to be an extension of the Minesweeeper project in Assignment 2. At this point, you should have an inference engine capable of taking clues from a minesweeper board, and performing inferences to determine information about the remaining squares on the board. (For the purpose of this assignment, we assume the total number of mines is unknown.) When there was nothing more that could be inferred, or there were no squares that were obviously safe to collect, the agent you built should've selected randomly from the available squares. For this assignment, we want to consider the problem of how to select the next square to click more intelligently - what is the best place to click next?

To consider this problem, we first need to devise an operational definition of 'best' - what are we actually hoping to accomplish? Using the paradigm from the first part of the assignment that clicking on a mine doesn't end the game (it simply incures a one point cost), we could quantify best in at least one of two ways:

- **Minimizing Cost:** The best choice of square is the one that minimizes the expected number of mines stepped on.
- **Minimizing Risk:** The best choice of square is the one that minimizes the expected number of squaress you have to unknowingly step on.

In the first case, this leads to a general trend to avoiding squares that are likely to be mines. In the second case, this leads to a general trend of stepping on squares that provide a lot of information about the future. In both cases though, we are dealing with decisions about an uncertain future, which will need to be quantified with probability.

Assesing the Future

Given a partially revealed minesweeper board, the previous project focused on figuring out if any squares were mines of safe. In this case however, we need to consider *how likely a square is to be a mine or safe.* To do this, we need to be able to assess the probability of a given square being a mine. This is easy to do wrong.

Consider the minefield in the following example, with the clues revealed and the unknown squares given by the variables *A,B,C,D,E,F:*

[2] [A] [1]

[B] [C] [D]

[E] [3] [F]

In the previous project, you implemented a system that should be able to determine that B is a mine and D is safe, but the rest of the board is undetermined. What, then, is the likelihood that A is a mine? Or that E is safe? It's tempting to argue that the probability that A is a mine is 2/3, based on the clue-value of w and there being three adjacent squares *A,B,C*. However, you would then also argue that the probability that A is a mine is 1/3, based on the clue-value of 1 and the three adjacent squares *A,C,D*. Given that B is a mine, you might even say further that the probability that A is a mine is actually 1/2, since one of A and C must be a mine to satisfy the clue-value 2. What is the truth?

If you work it out, you'll find that the only satisfying assignments are B is a mine, D as safe, and one of the three following combinations:

- C is a mine, A is not a mine, E is a mine, F is not a mine
- C is a mine, A is not a mine, E is not a mine, F is a mine
- C is not a mine, A is a mine, E is a mine, F is a mine

Because any one of these assignments is equally likely, we can then assess the probability of a given square being a mine as

[− − −] [1/3] [− − −]

[3/3] [2/3] [0/3]

[2/3] [− − −] [2/3]

From the above, we see that the least risky square to click on (in terms of stepping on a mine) is D, but the second least risky is in fact A.

In this way, given a partially revealed board, we can assess the possible states of unknown squares and in doing so determine the probability of any event. (For instance, the probability that A and F are the same thing is 2/3.)

Computational Comments:

- *You should be able to use your inference engine from Assignment 2 to help bootstrap the computation of these probabilities. How?*
- *Consider dividing the board into 'active unknowns', the squares that are touching some clue, and 'inactive unknowns', the squares that are touching no clues. For any inactive unknown, that square is equally likely to be a mine as it is to be safe. Why?*

In your writeup, describe in detail how you compute these probabilities and what you utilized from Assignment 2 to do so. If you approximate or estimate anything (not strictly necessary), be clear as to what you approximated, why, and why you feel it was valid.

The Basic Agents

In each of the following subsections, you will compare your agent from Assignment 2 (Pure Random Decision Making) with a (slightly) improved agent.

Minimizing Cost

Consider the following slightly improved agent, built off your agent from Assignment 2:

- Given the current state of the board, use your inference engine from Assignment 2 to figure out as much as possible about the remaining squares.
- If any square can be identified as a mine, flag it to not select it in the future.
- If any square can be identified as safe, click on it, add the clue to your knowledge base, and continue to infer until nothing definite remains.
- Once nothing definite (mines or safe) can be determined, assess for each unrevealed square how likely it is to be a mine.
- Click on the square with the lowest probability of being a mine (breaking ties at random), add the result to your knowledge base.
- Repeat (restarting the inference loop) until the board is cleared.
- The final total cost is taken to be the total number of mines stepped on.

Generate data and plot ‘mine density’ vs ‘average cost’ for both the pure random decision agent from Assignment 2, and the basic cost minimizing agent above. How does the slightly improved agent compare? Why? How frequently is the ‘improved’ decision making necessary?

Minimizing Risk

Consider the following slightly improved agent, built off your agent from Assignment 2:

- Given the current state of the board, use your inference engine from Assignment 2 to figure out as much as possible about the remaining squares.
- If any square can be identified as a mine, flag it to not select it in the future.
- If any square can be identified as safe, click on it, add the clue to your knowledge base, and continue to infer until nothing definite remains.
- Once nothing definite (mines or safe) can be determined, assess for each unrevealed square how likely it is to be a mine.
- For each square, determine *the expected number of squares that can be worked out if you clicked on it:*
  - Let q be the likelihood that the square is a mine, and let R be the number of squares that you could work out if it is a mine.
  - let 1 - q be the likelihood that the square is safe, and let S be the number of squares that you could work out if it is safe.
  - The expected number of squares that could be worked out is qR + (1 - q)S.
- Click on the square with the highest expected number of solvable squares (breaking ties at random), add the result to your knowledge base.
- Repeat (restarting the inference loop) until the board is cleared.
- The final total cost is taken to be the total number of squares stepped on *without knowing what they are.*

Generate data and plot ‘mine density’ vs ‘average risk’ for both the pure random decision agent from Assignment 2, and the basic risk minimizing agent above. How does the slightly improved agent compare? Why? How frequently is the ‘improved’ decision making necessary?

Bonus

How does the risk minimizing agent compare to the cost minimizing agent when it comes to minimizing the cost? How does the cost minimizing agent compare to the risk minimizing agent when it comes to minimizing risk? Generate data. Discuss. (Points for clarity and thourougness.)

Improved Agent

For one of either minimizing cost or minimizing risk, build an improved decision making agent that beats the simple agent described in the previous section (that isn’t just the other simple agent!). Plot the relevant data for your original Assignment 2 agent, the slightly improved agent (from the previous section), and your improved agent. Describe in detail, and justify, the logic and construction of your agent. Is your improved agent significantly better than the other two? How frequently does your improved agent make different decisions than the slightly improved agent? Is your improved agent as good as it can be? How could it be improved?

Bonus

Construct and analyze an improved agent for the other metric you didn’t already do.

Computational Comment Answers

From assignment two we were able to use our inference agent to help bootstrap the computation of the probabilities. Thus, the agent for this assignment was implemented as outlined below.

Inference: **Improved Agent**

1. The model for the improved agent finds multiple clues at once as long as the agent sufficiently finds the clues based on the probability that we compute. To do that, our sequence of tiles requires sufficient enough tiles (usually more than one or two) preferring to not engage in situations where tiles are next to each other but rather explore the one layer out from the adjacent tiles of the tile that was initially picked.
   (a) We used the inference engine, called probability_inference, which will compute the probability for neighboring tiles to declare which one might be possibly a mine, hidden, or cleared tiles.
2. Once the condition is satisfied, the improved agent starts exploring what can be possibly mines by comparing the probability of potential mine location. Also, the agent can declare revealed or hidden clear tiles by finding the lowest probability around the tiles that are revealed. Revealed clear tiles can be flagged and visited by the agent and the agent also declares the visited tile's number of mines that are surrounding it.
3. Not necessarily deducing everything, but, as long as the agent satisfies the condition for revealing multiple tiles that are safe enough to reveal, the agent continuously deduces the tiles and keep adding the clues into the queue.

Considering dividing the board into 'active unknowns' (*the squares that are touching some clue*) and 'inactive unknowns (*the squares that are touching no clues*), then we can infer that for any inactive unknown, that square is equally likely to be a mine as it is to be safe. The reason for this is because

Minimum Cost Visualization

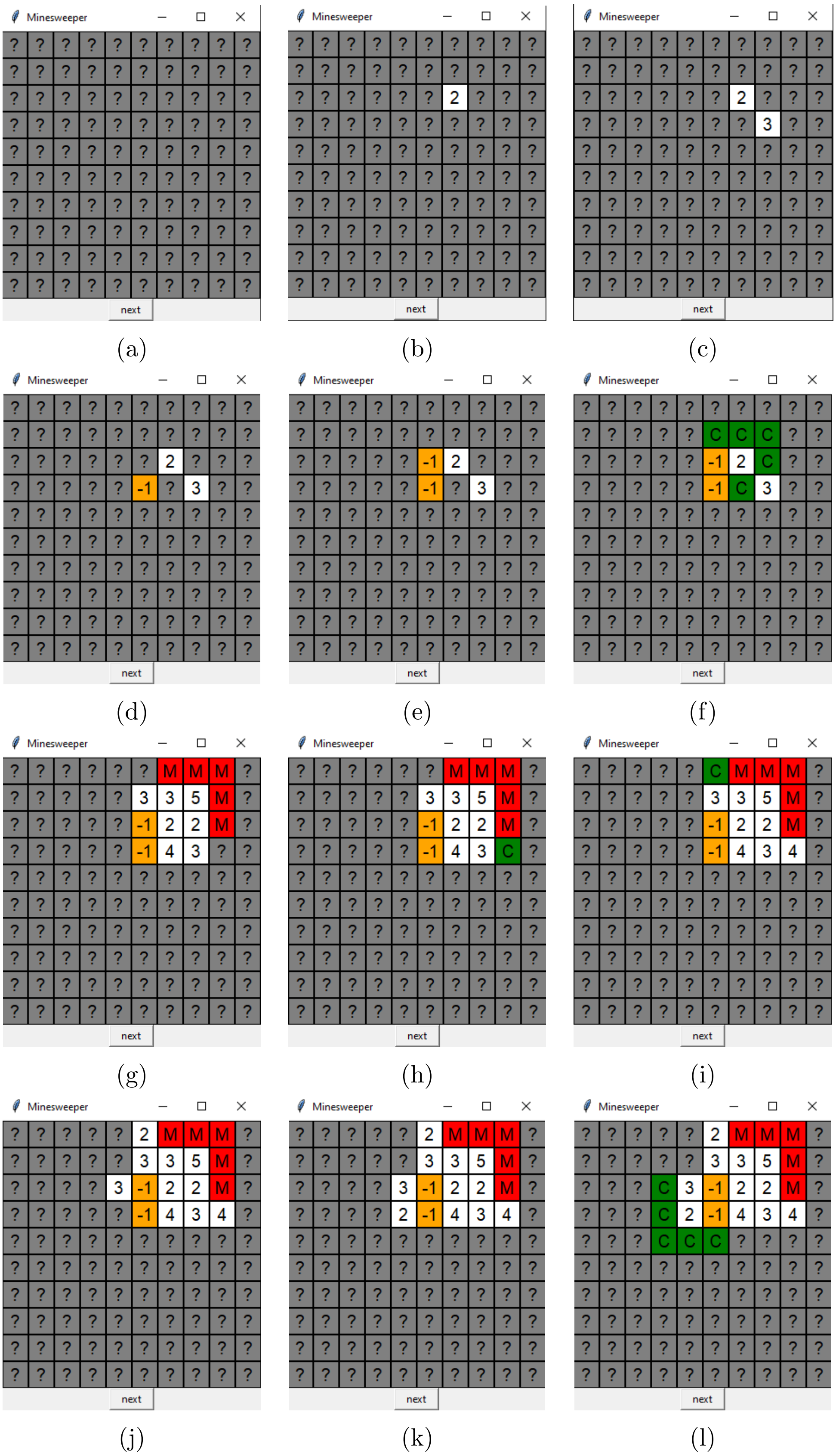
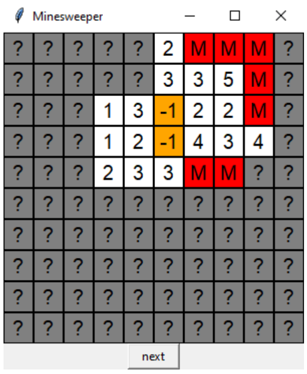
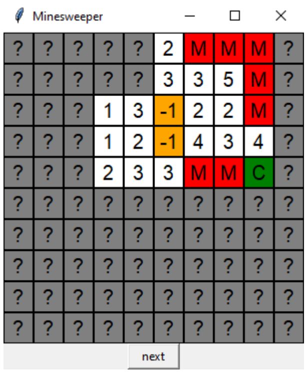


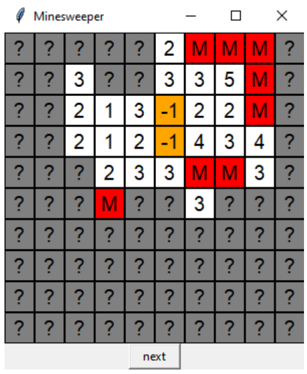
Figure 1: Minimizing Cost Agent Visualization



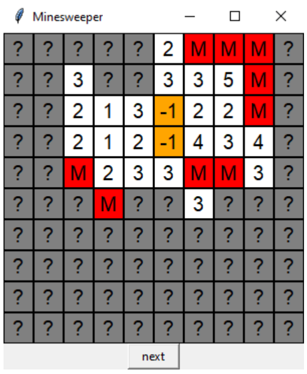
(m)



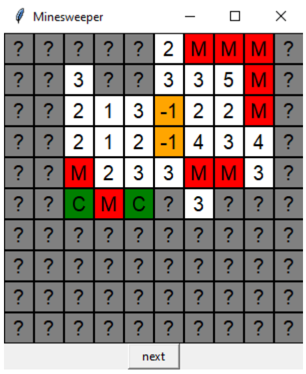
(n)



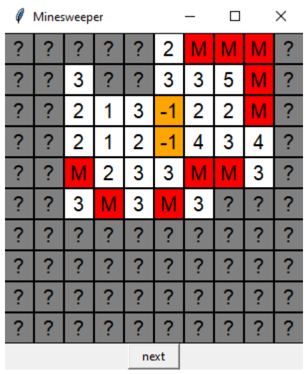
(o)



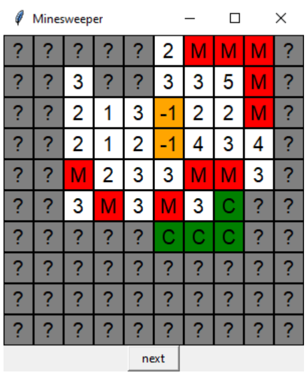
(p)



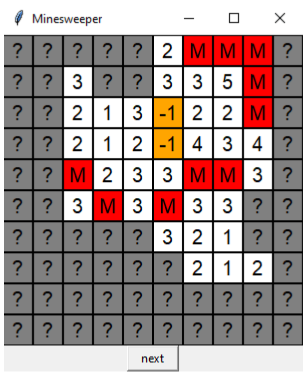
(q)



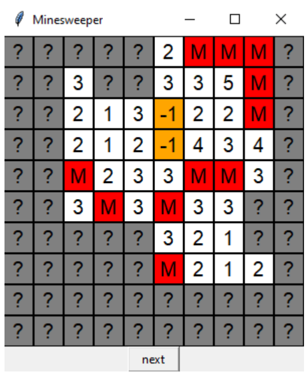
(r)



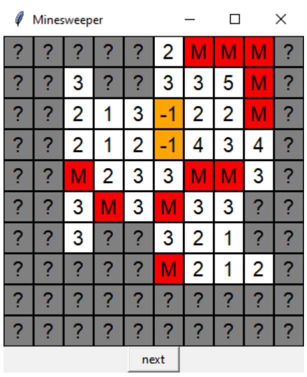
(s)



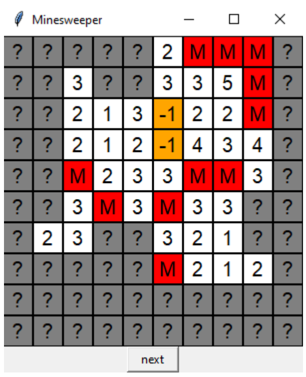
(t)



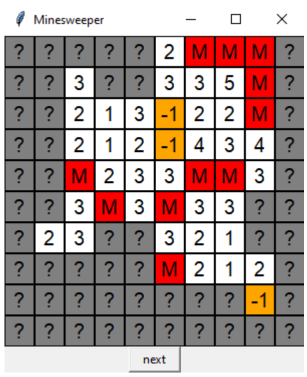
(u)



(v)

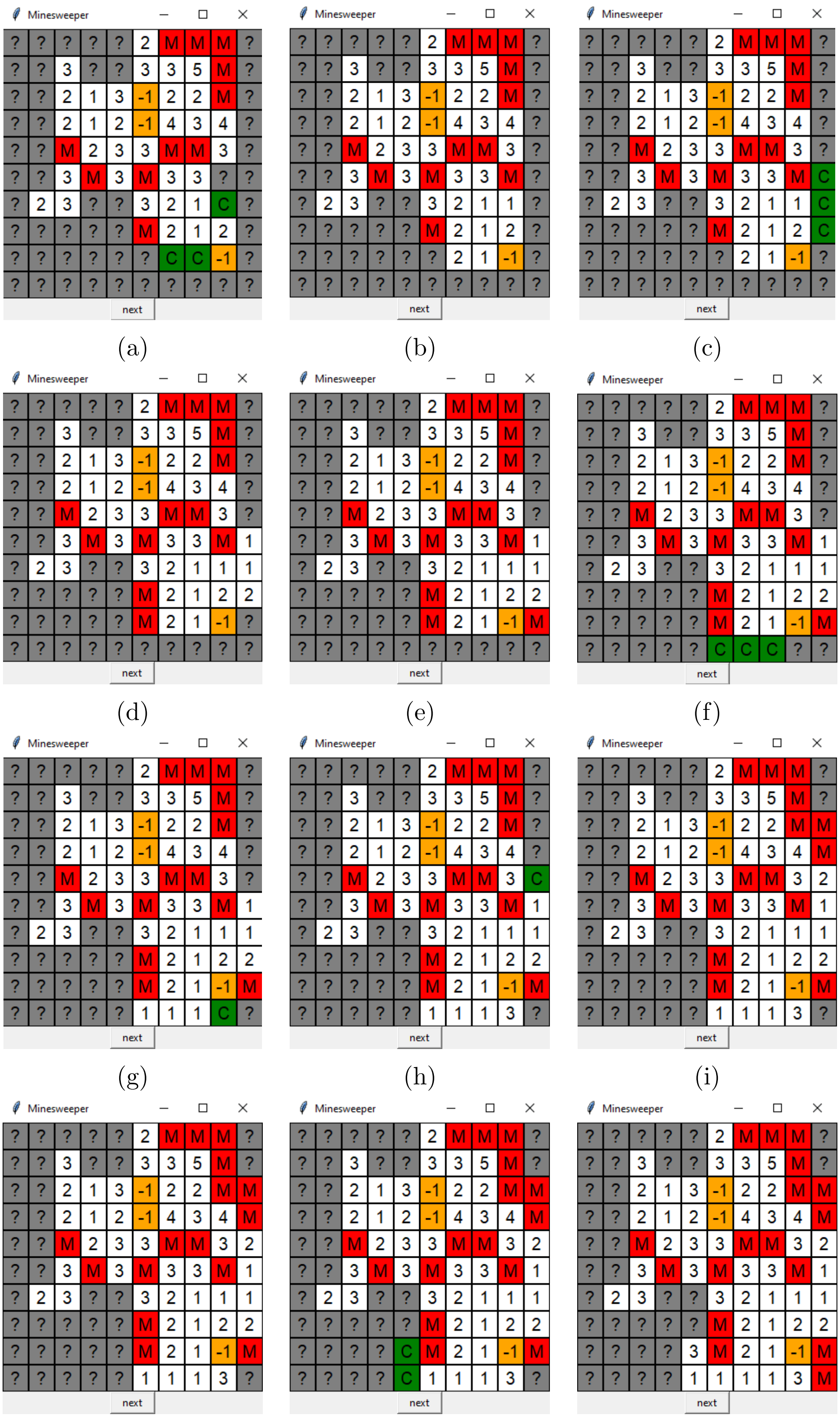


(w)



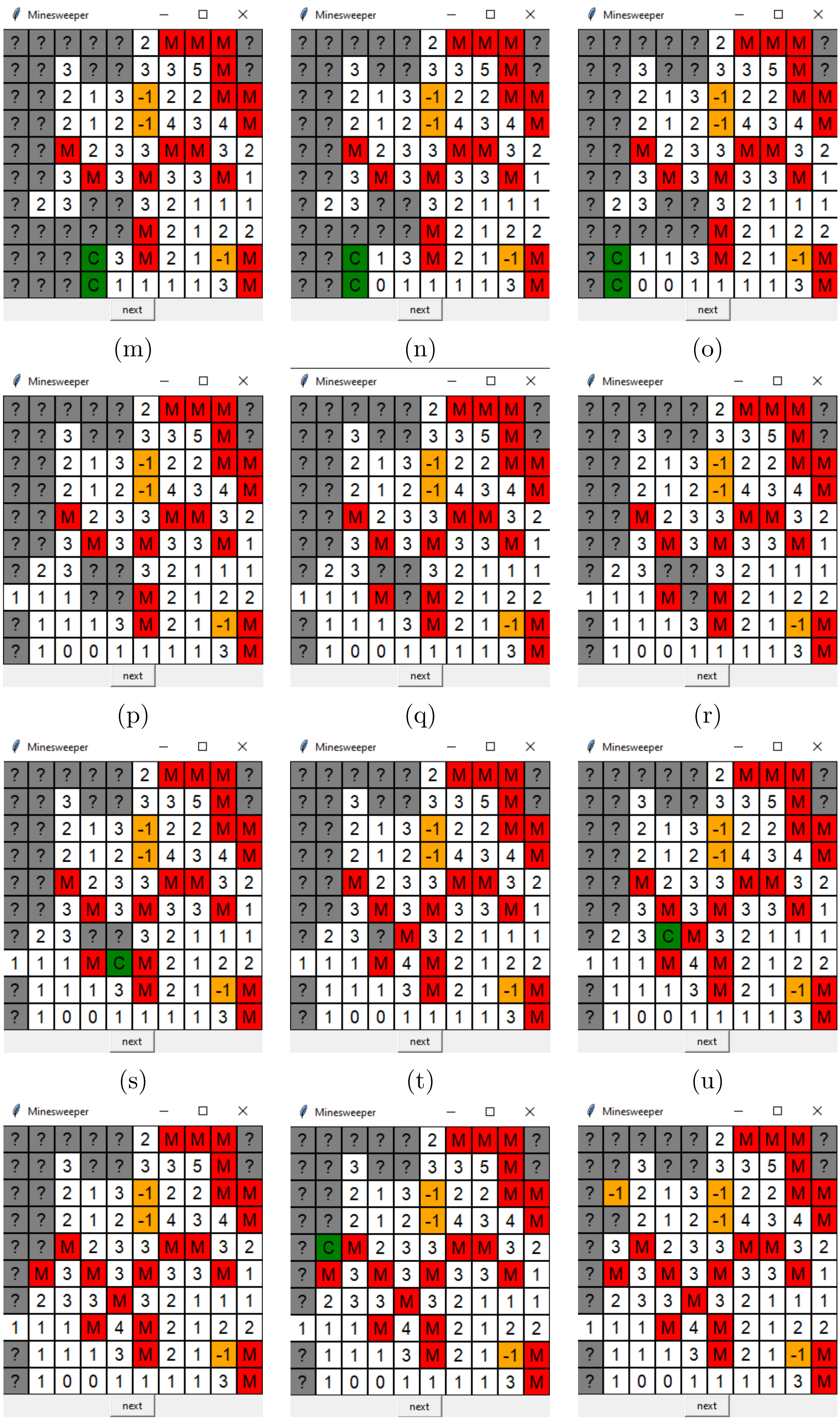
(x)

Figure 1: *Continued...*



(a) (b) (c)
(d) (e) (f)
(g) (h) (i)
(j) (k) (l)

Figure 2: *Continued...*



(m) (n) (o)

(p) (q) (r)

(s) (t) (u)

(v) (w) (x)

Figure 2: *Continued...*

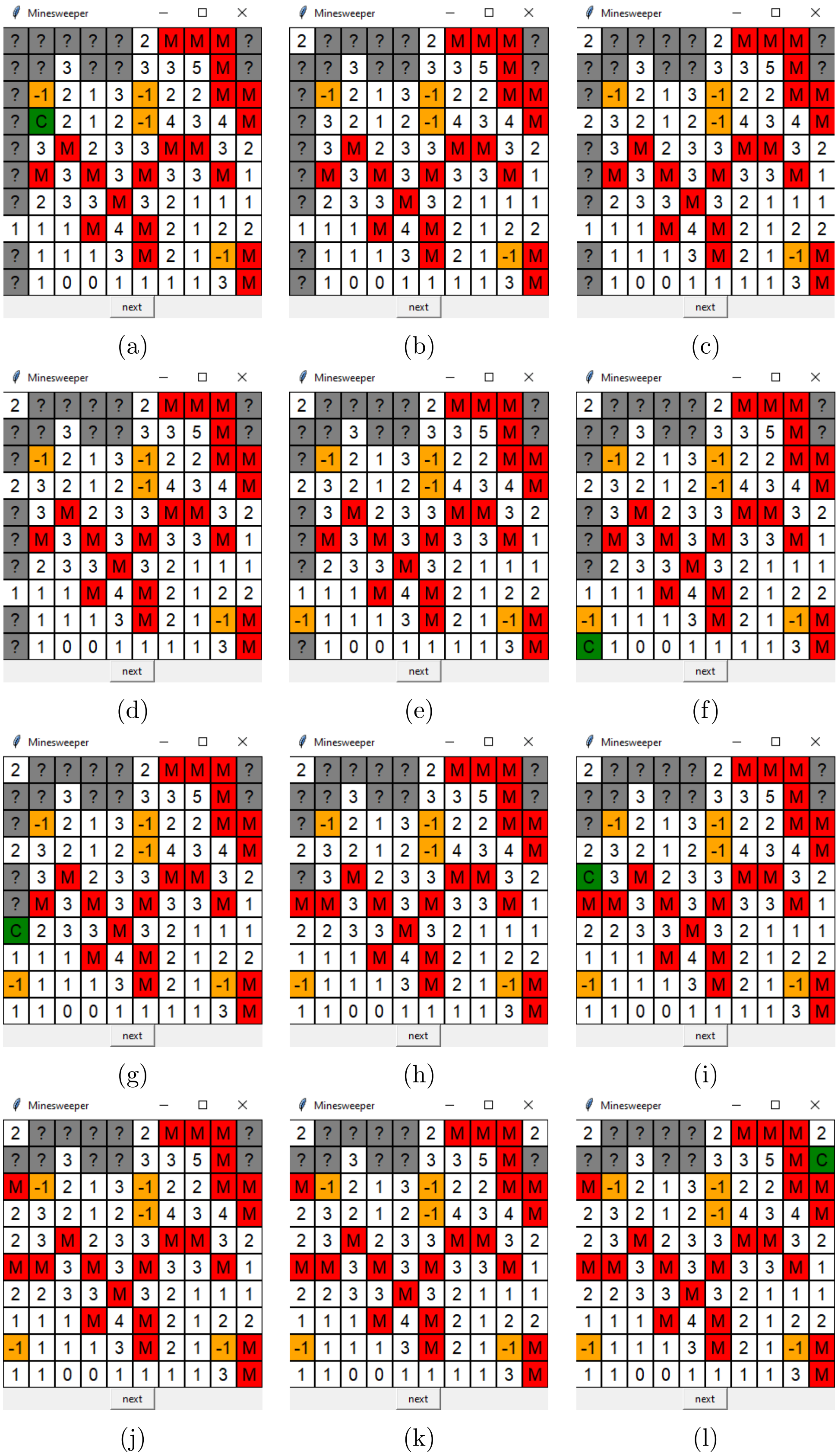
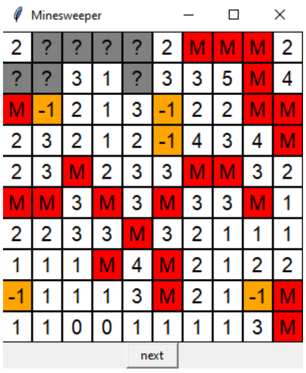
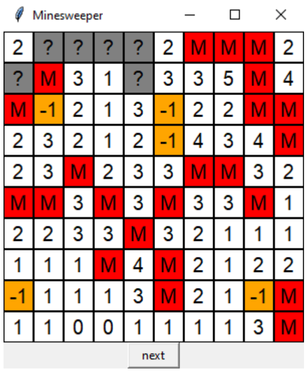


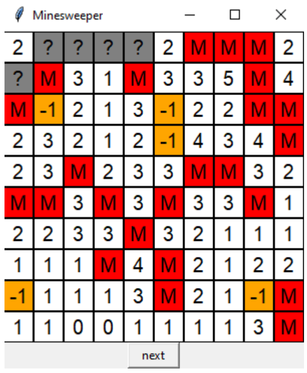
Figure 3: *Continued...*



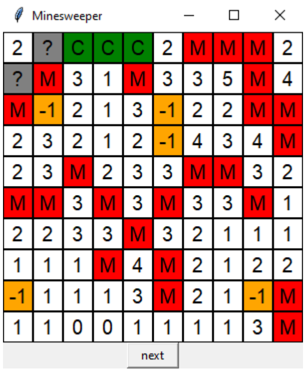
(m)



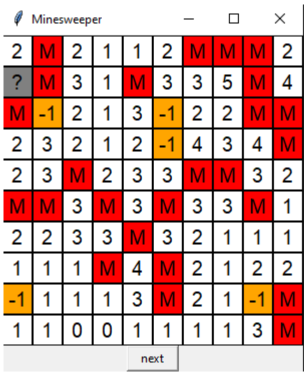
(n)



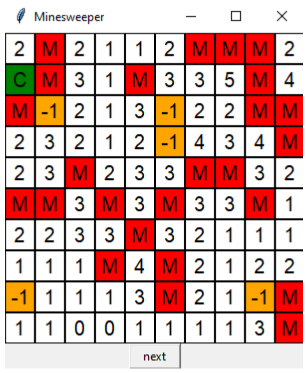
(o)



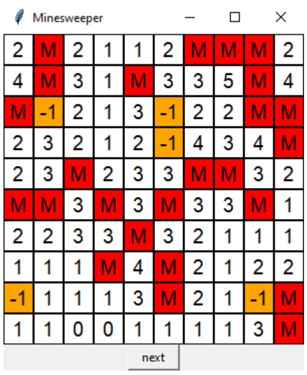
(p)



(q)



(r)

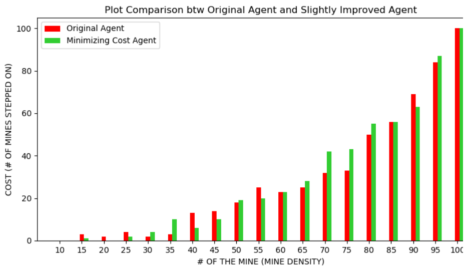


(s)

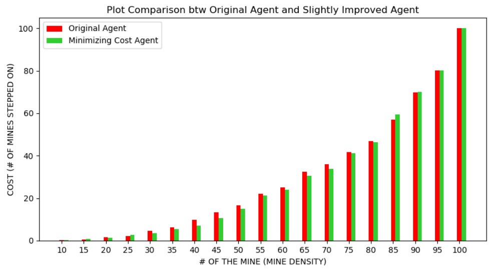
Figure 3: Minimizing Cost Agent Visualization
Grid:[10 x 10] **Mines**:[30]
Cost = total # of mines stepped on: 5

Minimum Cost Question Responses

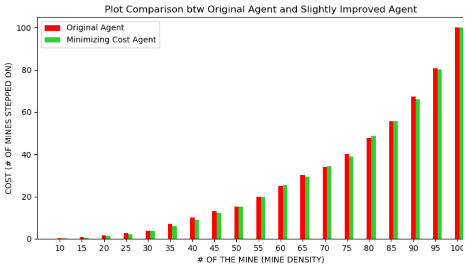
The plot(s) that are below (Figure: 4) is data for both the pure random decision agent from Assignment 2, and the basic cost minimizing agent.The original agent's data is denoted as the red bar graph(s), while the minimizing cost agent is denoted as the green bar graph(s). The plot was generated by calling the ***iterateForComparison()*** method in the ***minimizing_cost_agent.py*** file. This method was passed a 10 x 10 grid, with ten mines for varying games / iterations and the average cost was calculated for increasing mine densities. These one thousand games produced the Figure 4f). As one can see from the data, the original pure random decision agent from assignment 2, had a higher average cost for almost all mine densities.The only mine density that seemed to be an outlier for this plot in terms of average cost, was at mine density of ninety. However, when the mine density was one-hundred then both where neck and neck.(*This is to be expected given the fact that the whole board is covered in mines!*) From these results, we can then conlude that overall, the minimizing agent does 'slightly' better than the original agent overall. This is due to the fact that the minimizing cost agent calculates more accurately which neighbors, and the neighbors of those neighbors are most likely to be mines therefore avoiding those cells. Hence, the frequency for which the minimizing cost agent makes decisions is greater than that of the original agent. To quantify this frequency, seventeen out of the nineteen desities had the minimizing cost agent with a lower average cost than when compared to the original agent or around ninety percent. However, there are still some ambiguities left on this concept to quantify the cost for each of the agents because the data plot would look relatively different based on how we set the rule. For example, if we exclude the cost counting that happened from the initial random discovering square, then the plot would instead look hyperbolic for the cost-minimizing agent.



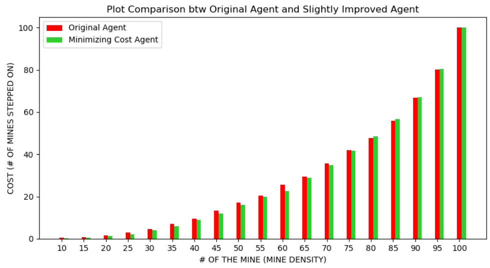
(a) 1 iteration(s)



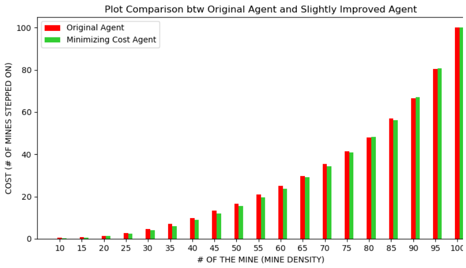
(b) 10 iterations



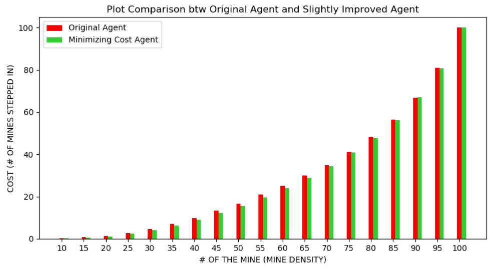
(c) 50 iterations



(d) 100 iterations



(e) 500 iterations



(f) 1000 iterations

Figure 4: Mine Density vs. Average Cost

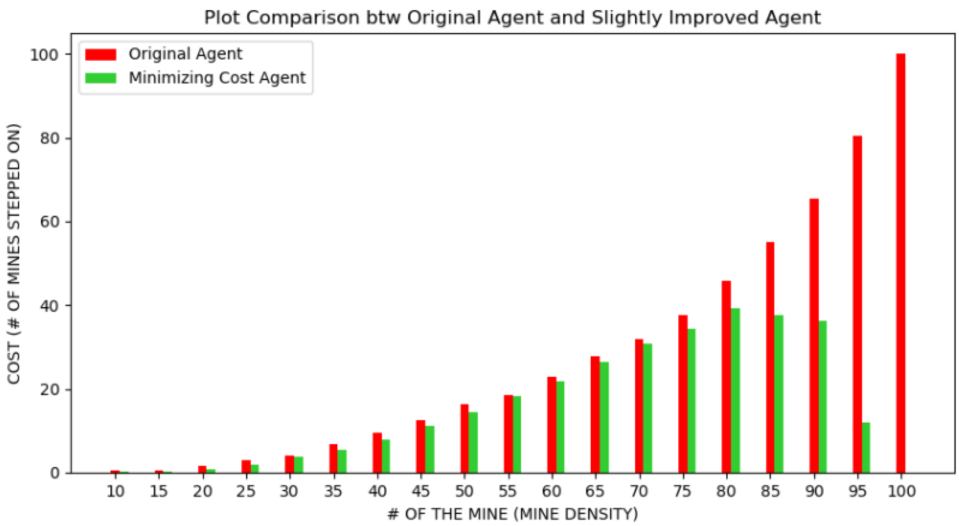
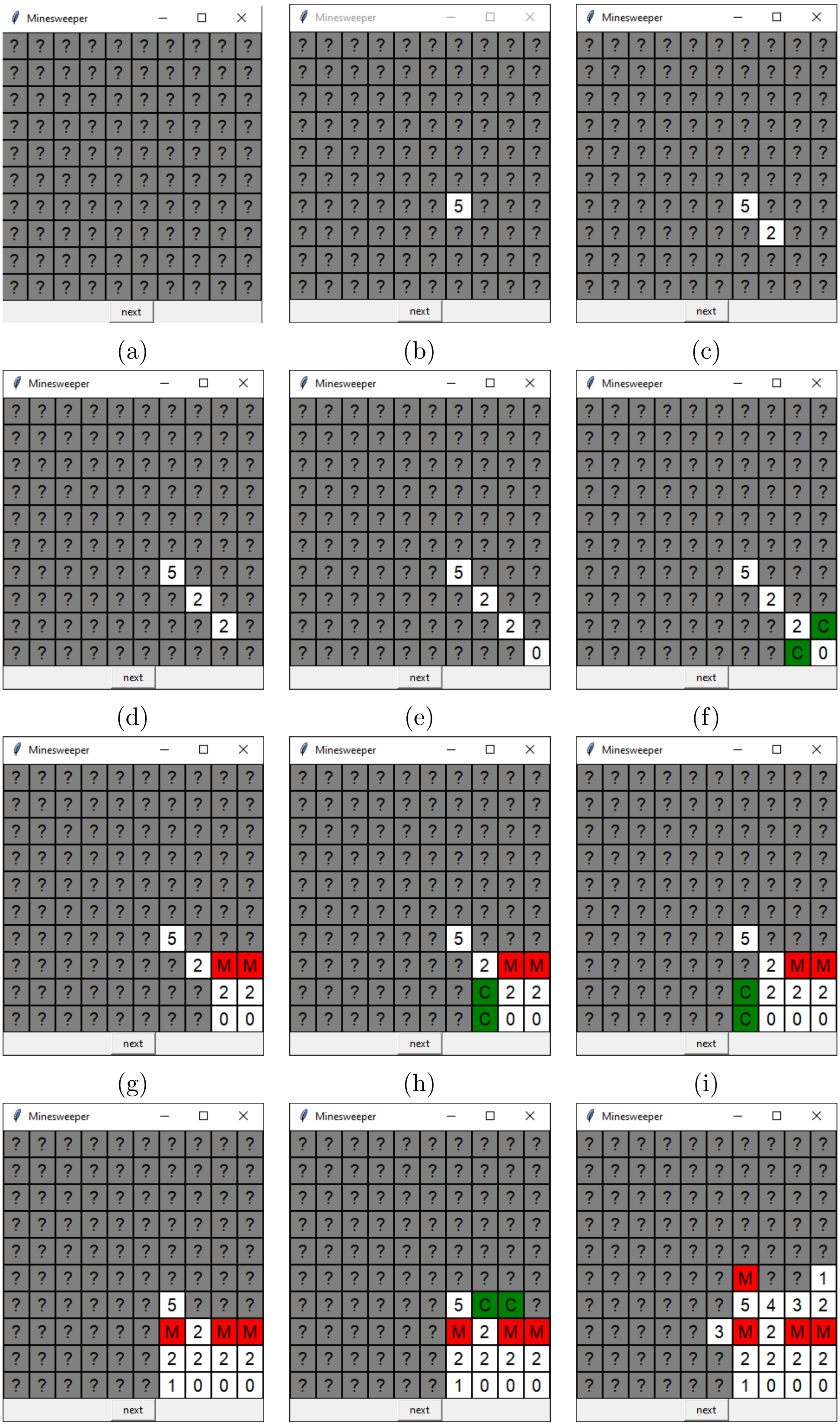


Figure 5: Mine Density vs. Average Cost
Random decision did not count towards cost

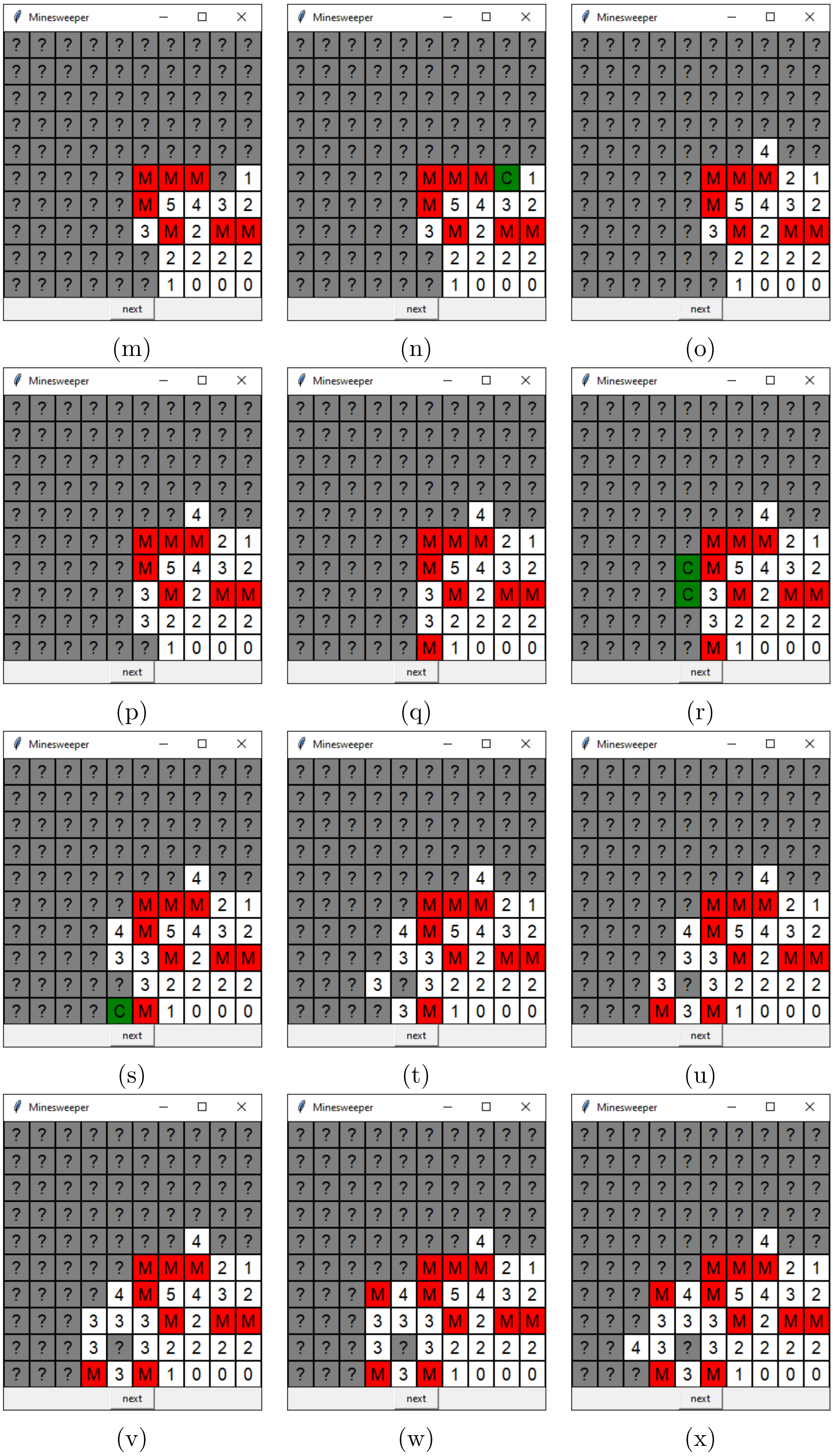
Figure: 4 and 5, demonstrates how the graphs output is highly dependent on the number of iterations. With more iterations, variances in the graph smooth over and become more consistent. Again it is a 10 x 10 grid, starting off with ten mines

Minimum Risk Visualization



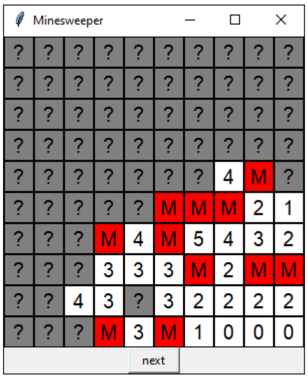
(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l)

Figure 6: Minimizing Risk Agent Visualization

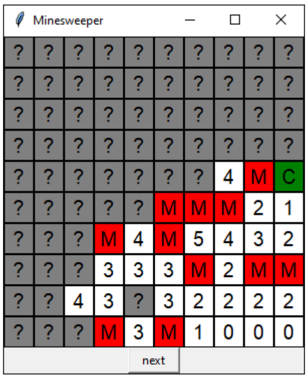


(m) (n) (o)
(p) (q) (r)
(s) (t) (u)
(v) (w) (x)

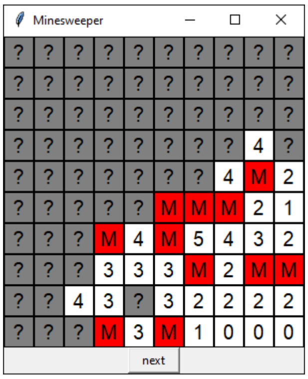
Figure 6: *Continued...*



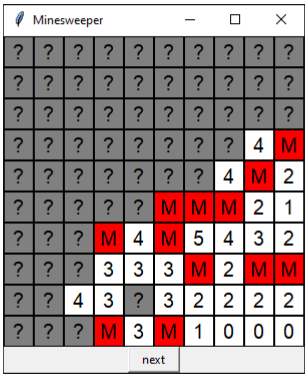
(a)



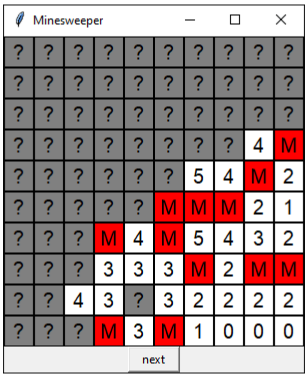
(b)



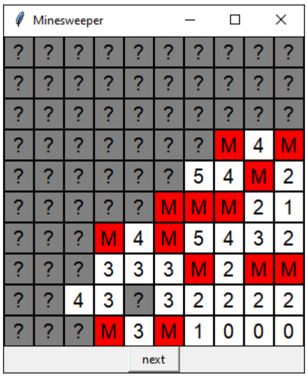
(c)



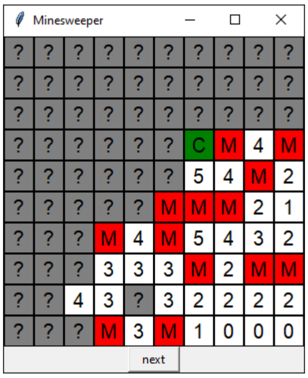
(d)



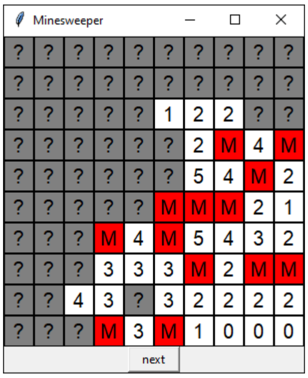
(e)



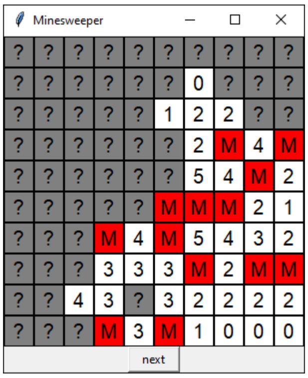
(f)



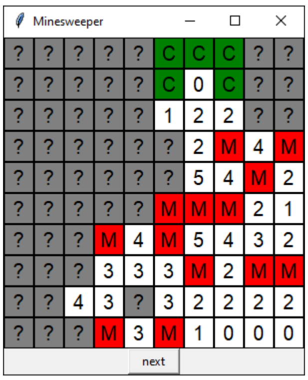
(g)



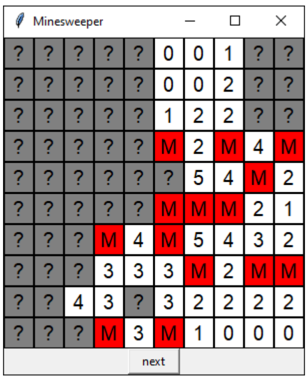
(h)



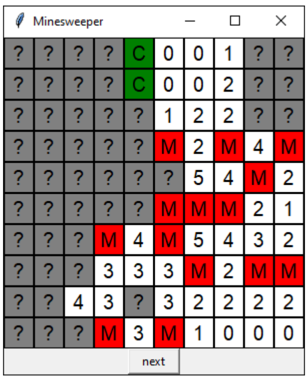
(i)



(j)

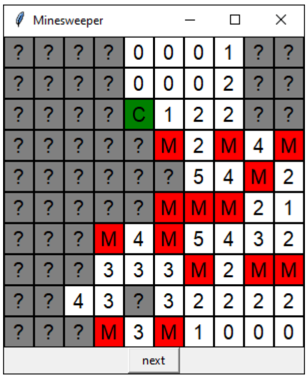


(k)

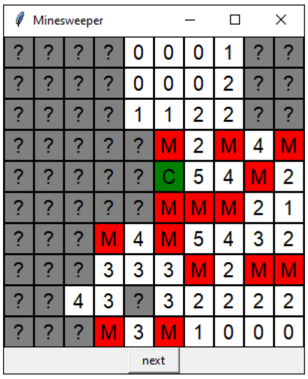


(l)

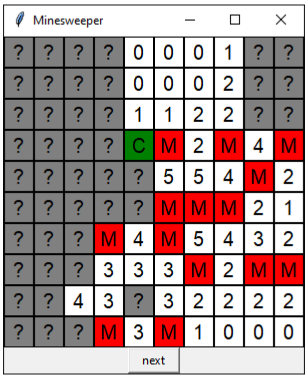
Figure 7: *Continued...*



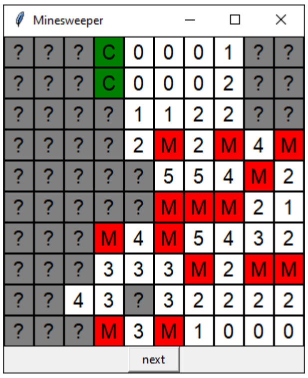
(m)



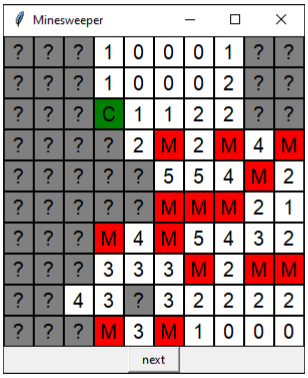
(n)



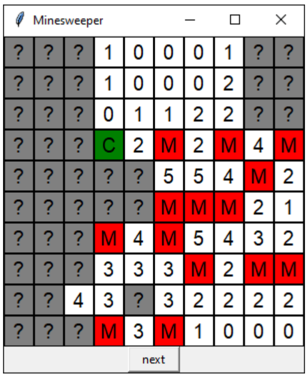
(o)



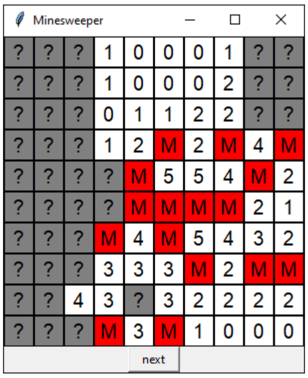
(p)



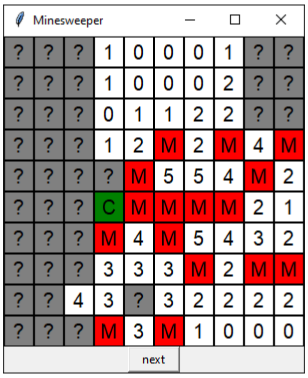
(q)



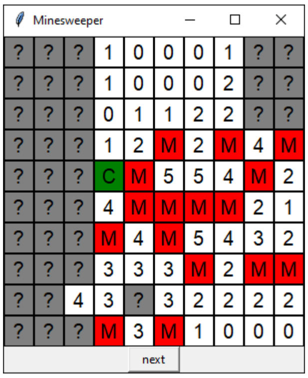
(r)



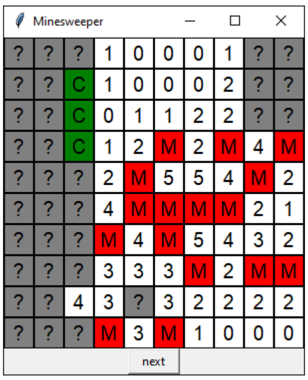
(s)



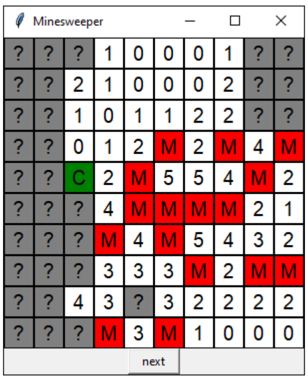
(t)



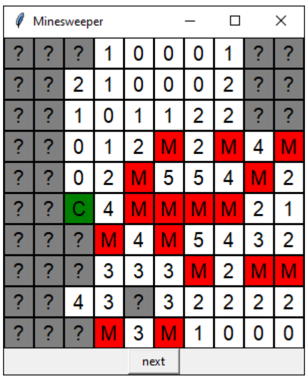
(u)



(v)



(w)



(x)

Figure 7: *Continued...*

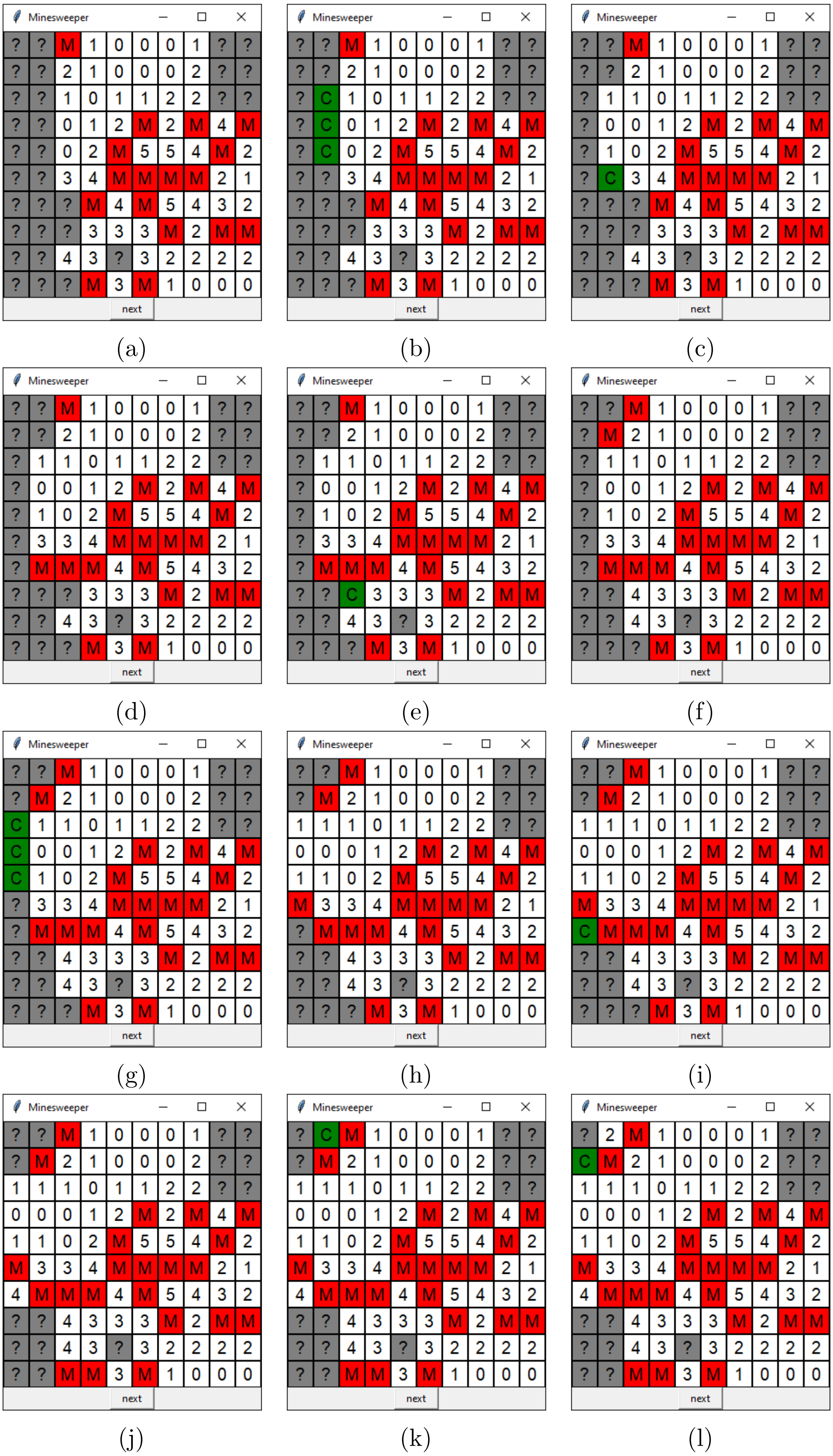


Figure 8: *Continued...*

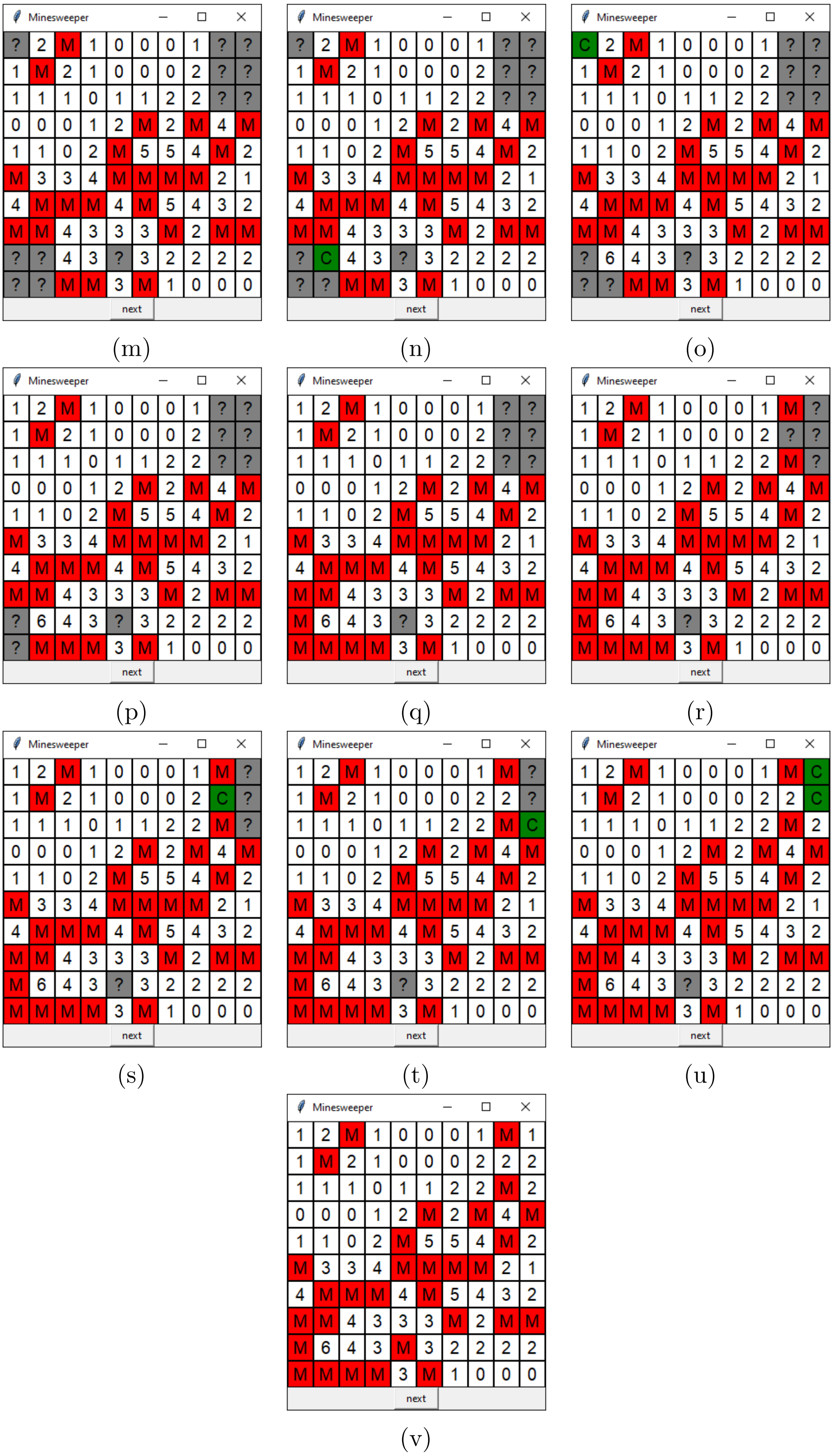
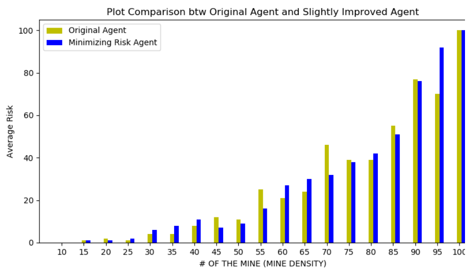


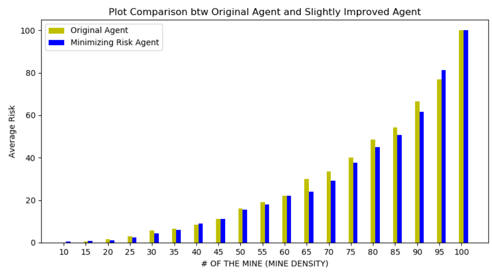
Figure 8: Minimizing Risk Agent Visualization: **Grid**:[10 x 10] **Mines**:[30]

Minimum Risk Question Responses

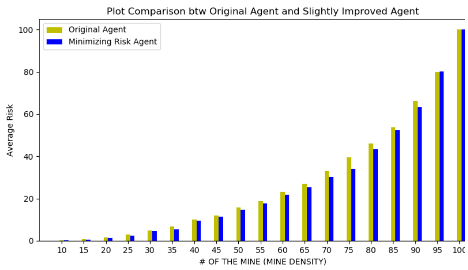
The plot(s) that are below (Figure: 9) is data for both the pure random decision agent from assignment 2, and the basic risk minimizing agent. The original agent's data is denoted as the yellow bar graph(s), while the minimizing risk agent is denoted as the blue bar graph(s). The graph was generated by calling the ***iterateForComparison()*** method in the ***minimizing_risk_agent.py*** file (*Line#: 583*). This method was passed a 10 x 10 grid, with ten mines where for varying games / iterations, the average risk was calculated for increasing mine densities. (*Mines increased by a factor of five*). The one thousand games / iterations produced Figure 9f. When analyzing the results from Figure 9f, one can see that for the overall the average risk was higher for the Original Agent, than that of the minimizing risk agent. In fact, the minimizing risk agent was efficient most of the time but the difference was ***not*** by a landslide by any means. Again, referencing Figure 9f, the only outlier mine density where both the original agent and the minimizing risk agent performed similarly in average risk was at a mine density of twenty.You can try to make the argument that for a density of one hundred they also perfomed similarly, but this happens to be the case for all board sizes. (*Which is expected given that all **n** x **n*** cells contain mines!) Anyways, the reason that the minimizing risk agent is more efficient for minimizing cost is because the agent considers each hidden tile as either a mine or not a mine by utilizing the utility function. Then using the implemented inferencing system, it calculates the maximum number of squares able to be solved for each of the hidden tiles and chooses the maximum. Hence, it was frequently figuring out more information than the original agent. Additionally, if we wanted to quantify how frequently 'improved' decision making was necessary, then it would be around eighty-four percent of the time (*sixteen out of the total nineteen mine densities had the minimizing risk agent with a lower average risk.*).



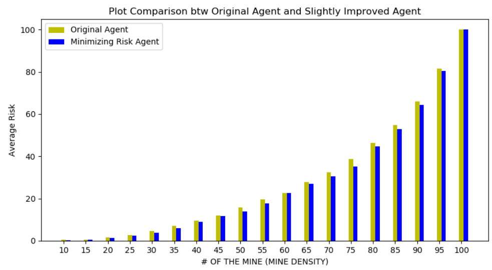
(a) 1 iteration(s)



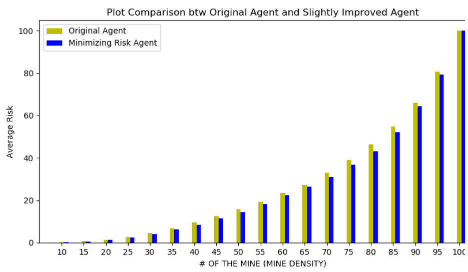
(b) 10 iterations



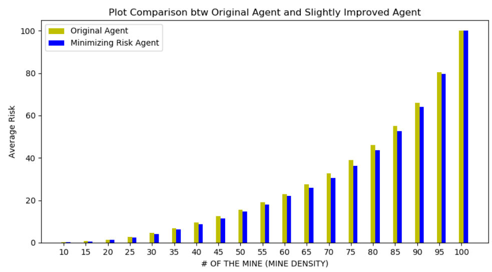
(c) 50 iterations



(d) 100 iterations



(e) 500 iterations



(f) 1000 iterations

Figure 9: Mine Density vs. Average Risk

Figure: 9, demonstrates how the graphs output is highly dependent on the number of iterations. With more iterations, variances in the graph smooth over and become more consistent. Again it is a 10 x 10 grid, starting off with ten mines

Bonus Answer

Now that we have pitted the two different improved agents (*Minimizing Cost and Minimizing Risk agents*), against the pure random decision agent, how do the two compare when compared to each other? In order to both answer and see how the risk minimizing agent compares to the minimizing cost agent, we gathered data for both of them, and put them to the test. To give more background, both agents were iterated one thousand times / games on a 10 x 10 board, starting with ten mines.

The first question one can ask is how both improved agents would perform when one wanted to test for minimizing cost? Intuitively, one would expect for the minimizing cost agent to outperform the minimizing risk agent, since that is its' specialty. Well, after the one thousand iterations we got figure 10. According to the data from that plot, it seems that in actuallity, the minimzing risk agent performed better in terms of having a lower average cost. Although, the gap was not 'large' by any means, it is still very noticable, when compared to the cost differences from Figure: 4.

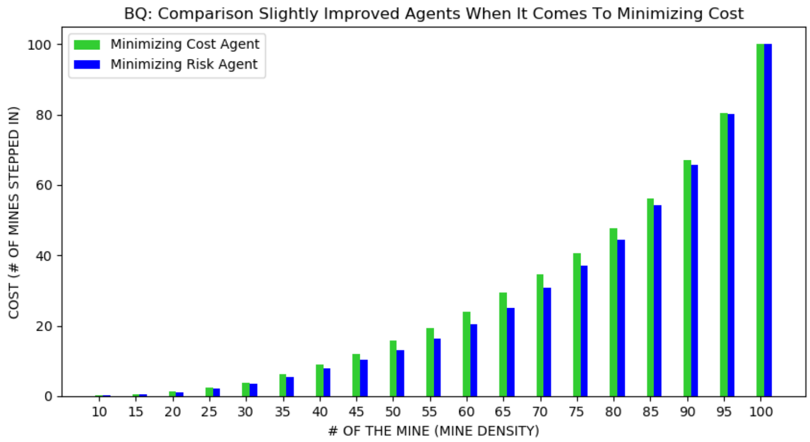


Figure 10: **Minimizing Cost Agent vs. Minimizing Risk Agent** (*Minimizing Cost Comparison*)

The second question one can ask is the reverse of the first question. That is to say, how would both the improved agents perform when it comes to minimizing risk. Again, intuitively one would expect for the minimizing risk agent to perform better than the minimizing cost agent. *But*, is that the case? Well again, we ran through one thousand iterations and generated a plot to answer this question. The resulting plot can be seen in Figure: 11. As one can observe, the minimizing risk agent again outperforms the minimizing cost agent when it comes to average risk. This time the difference in average risk is more noticeable than that of Figure: 9. The conclusion that we can come to from these two comparisons *minimizing cost and minimizing risk*, is that our implementation of the minimizing risk agent is better for both of those said cases, if we really cared about doing the best that we could. Hence, our implementation of the minimizing risk agent is more well adapted to handle both of those cases.

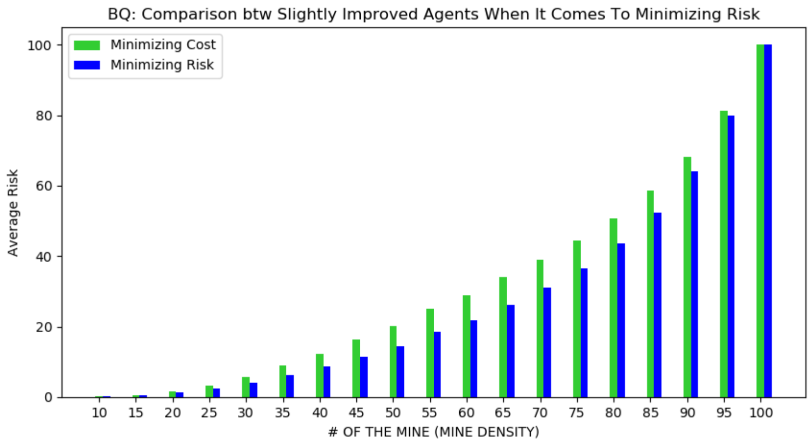
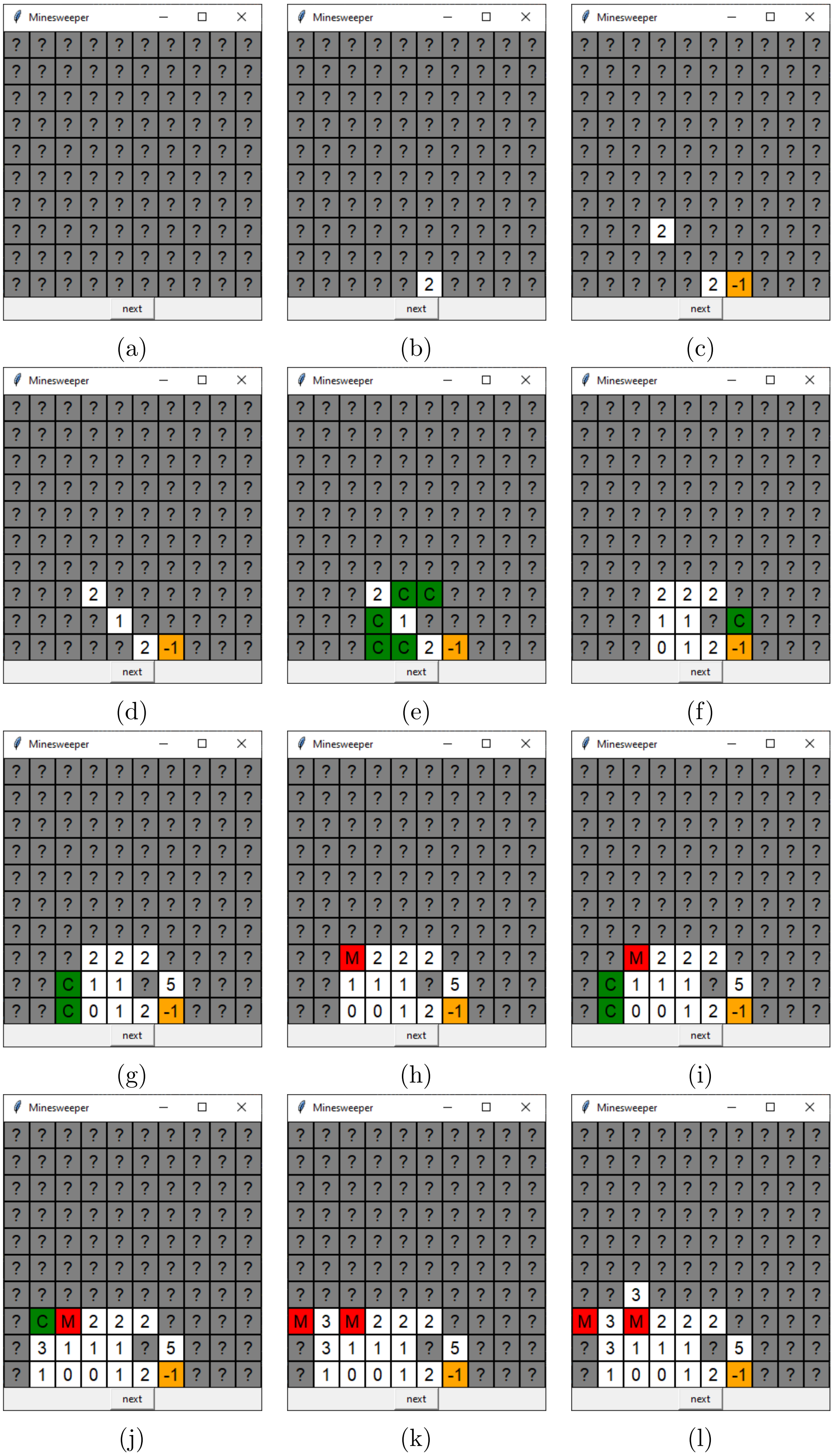


Figure 11: **Minimizing Cost Agent vs. Minimizing Risk Agent** (*Minimizing Risk Comparison*)

Improved Agent Visualization



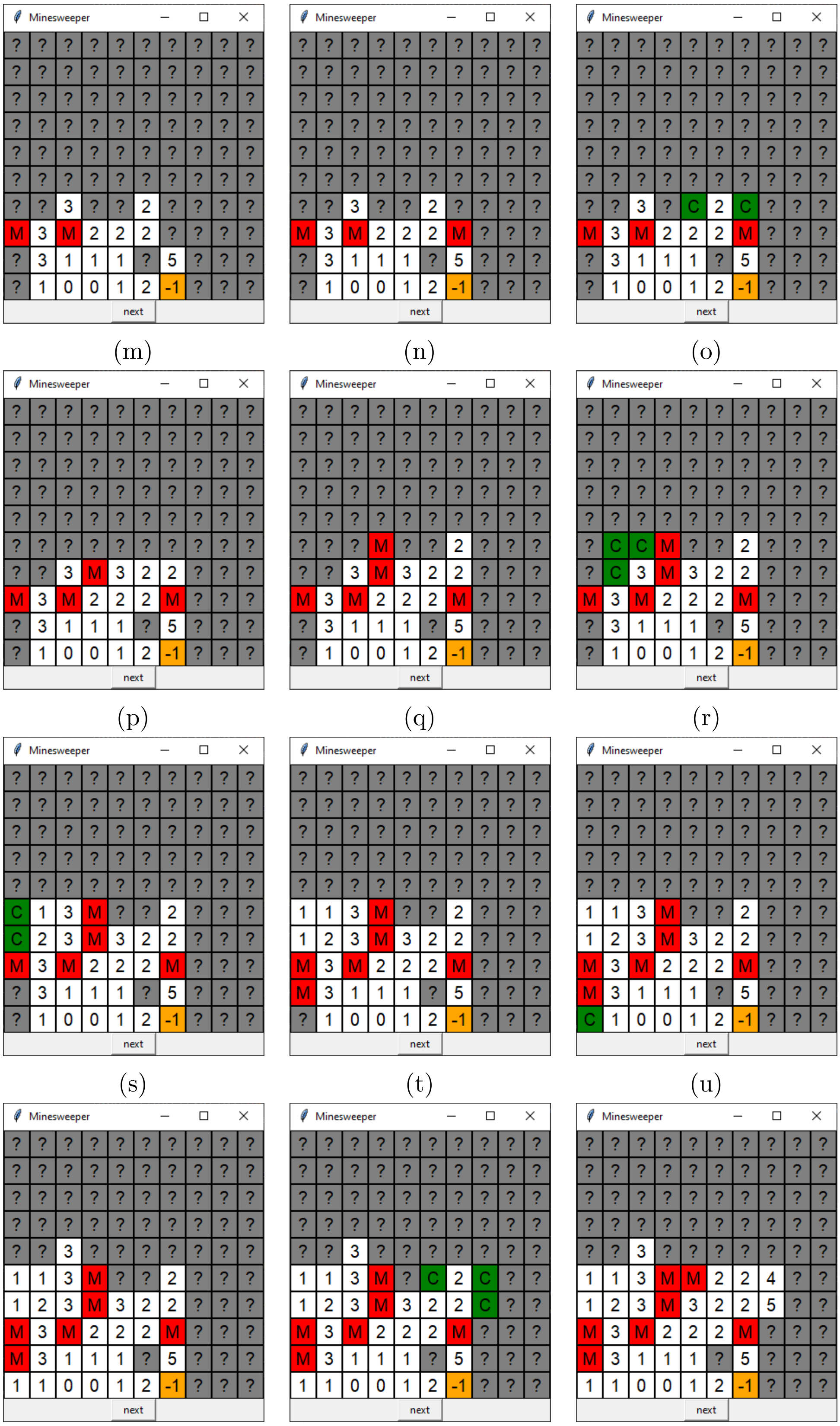
(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figure 12: Improved Agent Visualization



(m) (n) (o) (p) (q) (r) (s) (t) (u) (v) (w) (x)

Figure 12: Improved Agent Visualization

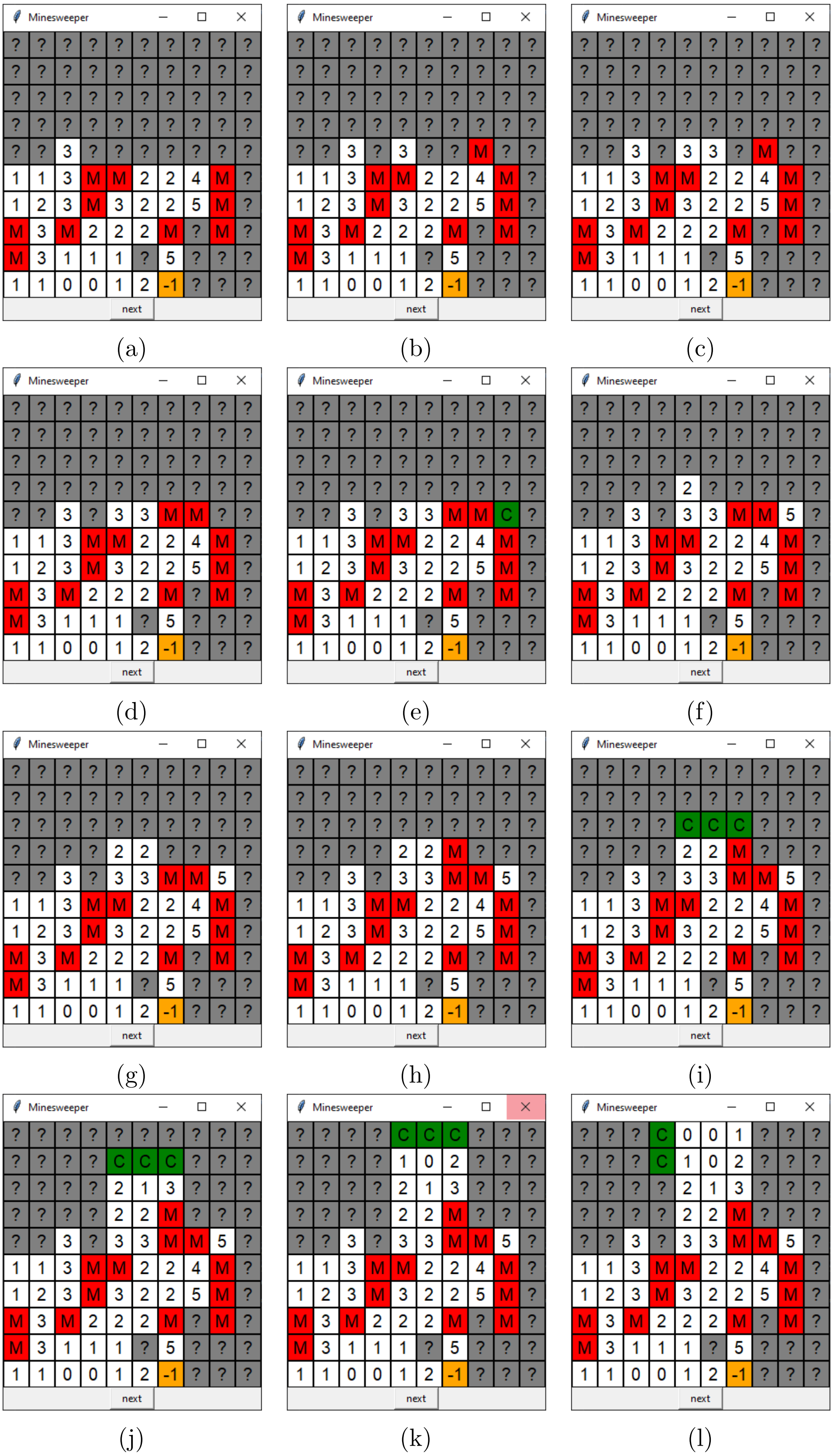


Figure 13: Improved Agent Visualization

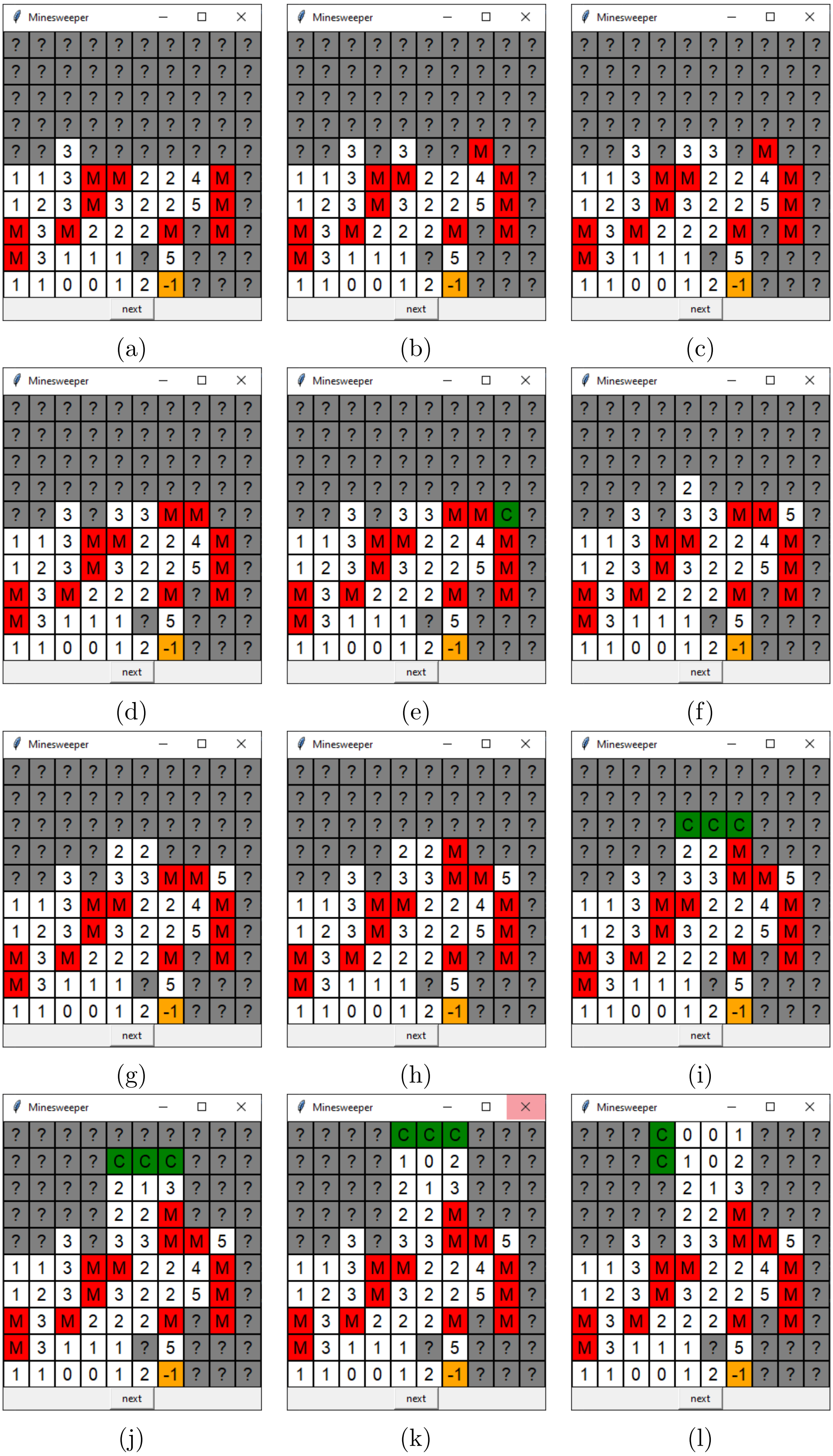


Figure 14: Improved Agent Visualization

Improved Agent

For one of either minimizing cost or minimizing risk agent, we built an improved decision making agent for minimizing cost. Thus, to see just how improved the agent did, it was tested against the original assignment 2 agent, and the slightly improved minimizing cost agent (from the previous section). To gather the relevant data, each of the three previously stated agents were played through one thousand games / iterations on a 10 x 10 board, and then plotted in Figure 15f below. The: red bar graph(s) represent the original agent from assignment 2, green bar graph(s) represent the slightly improved agent from the previous section and the blue bar graph(s) represent the improved agent.

The fundamental idea for the improved agent, is to come up with improved decision making by considering two steps ahead for the optimized decision in the future. A very critical point with this strategy, is that the improved agent should find every possible measure that affects future moves, which will ultimately reduce the probability of stepping on a mine.

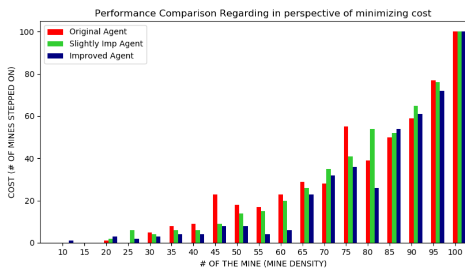
To implement this idea, we needed to combine three insights to come up with a plausible solution for this improved agent.

- First, with the clues that are found from the randomly and initially uncovered square, you need to find the next best move. in order to accomplish this task, computing of each of the clue's surrounding tiles was needed, and thus done by the risk and probability inferencing, which are implemented in the slightly improved agent. By doing this, the improved agent can compute all the possible moves and choose the best of those moves to uncover the next square.

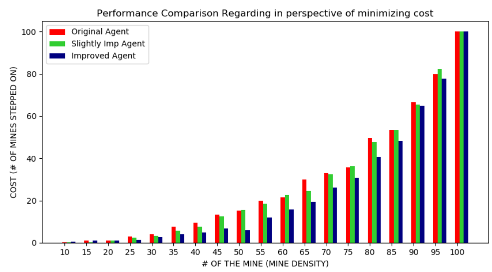
- Second, it is crucial that the risk inference be computed before the probability inference because risk inference will make the appropriate decision when and if there are enough clues to be sure whether exploring a certain tile is releveant to identifying if it is either safe or a mine.

- Third, once there are not enough neighboring tiles/squares or inferences that will result in discovering a new square, an improved random square selection will be made by discovering a tile that is one square away from the board of lowest probability inference of stepping on a mine. Additionally, this improved random square will be selected by checking the most relevant possible clues that are close to each other (*This improved random function is described on line# 532 in **ImprovedAgent.py***).

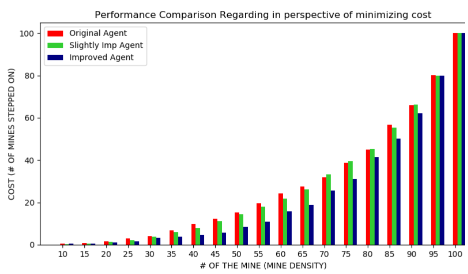
Once this strategy is implemented, the improved agent did an overall 'better' job at minimizing cost by stepping on less mines than the other two agents! Additionally, if we wanted to quantify how frequently 'improved' decision making was necessary, it would be around eighty percent of the time.(Fifteen out of the total nineteen mine densities had the improved agent with a lower average cost).



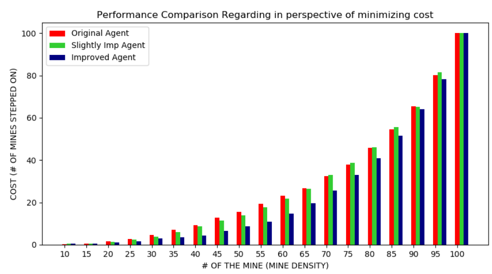
(a) 1 iteration(s)



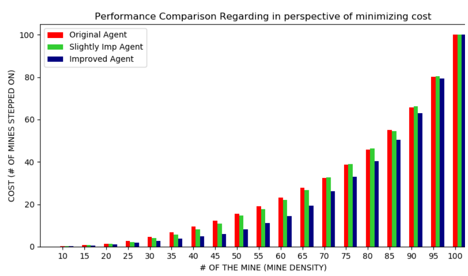
(b) 10 iterations



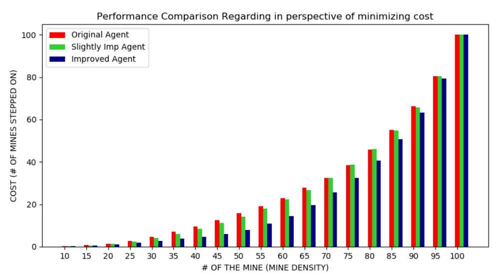
(c) 50 iterations



(d) 100 iterations



(e) 500 iterations



(f) 1000 iterations

Figure 15: Improved Agent Comparison
(*Minimizing Cost*)

Figure: 15, demonstrates how the graphs output is highly dependent on the number of iterations. With more iterations, variances in the graph smooth over and become more consistent. Again it is a 10 x 10 grid, starting off with *ten mines*

Based on Figure: 15f, it can be concluded that the current improved agent is good enough to justify that it makes decisions with better performance than either the original agent or the slightly improved agent. However, this highly depends on how it is implemented and tested out. The implementation depends on how the rules are constructed which essentially means that keeping count for ***only*** valid decisions is a factor that changes the plot. For example, we can skip counting the cost (*for stepping on a mine*) from the inital decision to uncover a randomly selected square/tile because the initial decision does not yet have control over whether or not the random tile will be a mine or not, (*lack of clues*), hence for this sake a tile is as likely to be a mine as it is not likely to be a mine. Therefore, from the perspective of adding more conditions and restrictions on determining what move adds to the cost, it will influence the plot highly than those of Figure: 15f because random discovery would not add to the cost in this case. To test this out we modified the *initial_random_outside()* function (Line#: 508 in ***ImprovedAgent.py***), and got the following plot, Figure: 16

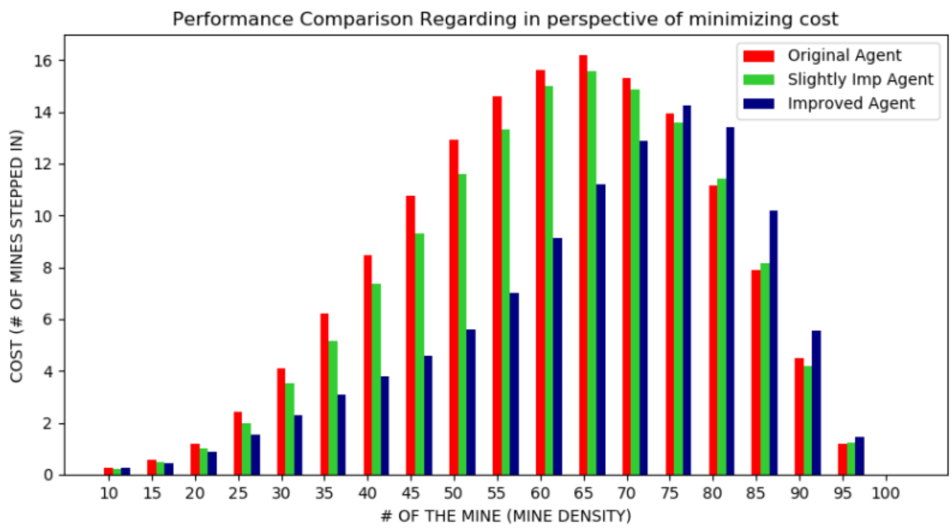


Figure 16: Improved Agent Comparison
Not counting random decision to the cost
(*Minimizing Cost*)

Figure: 16, was produced by passing a 10 x 10 grid, for one thousand games / iteration, with increasing mine densities.

As we can see from Figure: 16, when random decision is not taken into account for the cost, the results for the improved agent (*blue bar graph(s)*) is similar to the results of Figure: 15f, up to a mine density of twenty, but then we start to see a major change onwards. Again, refering to Figure: 16 from mine densities greater than twenty and less than or equal to seventy, the Improved agent vastly outperformed the other two agents. However, once the mine densities were greater than seventy (*not including a mine density of one hundred*), the average cost for the Improved agent was greater than that of the original agent and the slightly improved agent, because of the fact that the improved agent was looking more moves ahead on a grid mostly comprised of mines.

Contributions

Jae: All code for following files: *ImprovedAgent.py, ImprovedGame-setting.py, ImprovedVisualization.py, minimizing_cost_agent.py, minimizing_risk_agent.py*, and Improved Agent answer)

Cesar: Creation of LaTex doc & responses for minimizing cost / minimizing risk agents