Minesweeper 2.5

Ilana Zane, William Bidle, Rakshaa Ravishankar

April 12, 2020

1 Computation

1.1 Methodology

We utilized the advanced agent from Project 2 and built upon it. The agent from Project 2 would go through any obvious moves first, and then try to make any inferences it could for the board. If nothing at all could be determined from this step, it then needed to access the probabilities of our 'active unknowns,' which are cells that have some information tied to them (i.e. an adjacent cell has been revealed). To get these probabilities, we utilized the reduced matrix that our inference agent generated for us and created a list of potential solutions. We obtained the potential solutions by taking the dot product of our matrix with vectors of 1's and 0's (indicating a variable being safe (0) or a mine (1)) and checked whether or not the result was equal to our answer vector, which contains revealed clues. From these solutions, we determined the probabilities by averaging how often a variable appeared as a mine (see Figure 5).

1.2 Reducing Variables

In general, if we have n variables per equation and each of these are either safe (0), or a mine (1), then our program will generate 2^n possible solutions that we need to check. For larger boards, we reach a very unreasonable number of solutions that the computer will have to simulate. If our board was even 10x10, the 100 cells would result in 100 variables per equation, leading to 2^{100} potential solutions that we need to check. However, this number can be drastically reduced. We only need to simulate potential solutions that include our 'active unknowns,' and not include any 'inactive unknowns.' In our situation, inactive unknowns correspond to any 0 columns in our reduced matrix, which tells us that no information has been revealed about that cell and its neighbors. This reduces our potential solutions and we end up simulating a number of solutions within a reasonable amount of time (see Figure 3).

1.3 Approximation

Even after this reduction, there were still scenarios where equations have a significant amount of variables. The result would be a very slow run time due to a large amount of computations. We found that any situation with more than 18 variables per equation led to our program taking a long time to compute possible solutions for just one step in the game. In order to solve this problem, we set a limit to when our program would calculate exact probabilities and when it would begin to estimate. If our equations exceeded 18 variables (i.e. greater than 2^{18} possible vectors to check), we would only simulate from a random selection of 2^{18} vectors in the total amount. For example, if we had 20 variables, we would randomly select 2^{18} of the 2^{20} total vectors to then continue and find the dot

product for possible solutions. This would yield approximate probabilities, however since our sample size is relatively large, we would end up with values that were close to the exact ones. The figure below shows an example where we simulate the exact probability vs. the approximate probability for a system of equations containing 20 variables (the X-axis is the variable number and the Y-axis is probability value). Even though we are only looking at 2^{18} out of the 2^{20} vectors in our approximation, the probability results seen are close to the exact answer. In some places we can see that the approximate is even the same value as the exact, so not only would our approximation help with space/time constraints, but it would also keep accuracy in the results.

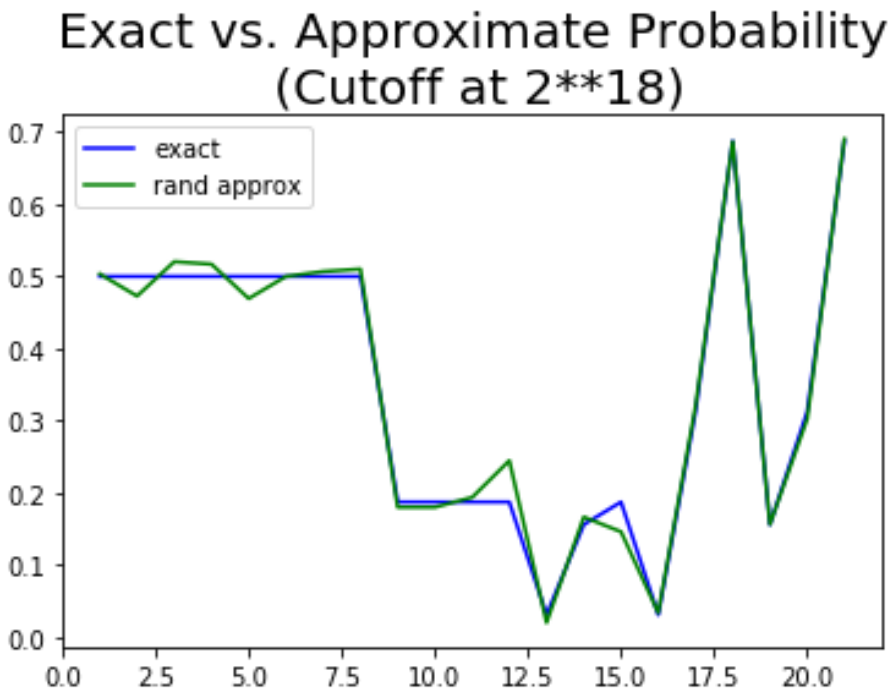

Figure 1

1.4 Example

1.4.1 Generating Solutions

We have created a step by step explanation as to how we calculated general probabilities using the example provided in the Project 2.5 Assignment PDF.

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

Figure 2

Based on the given variables, our inference agent creates a system of linear equations (our augmented matrix) and the associated answer vector that contains all clue values.

Immediately our inference agent determines that cell B is a mine and D is safe. The equations are created by looking at each uncovered cell and adding its clue value into our answer vector. In the assignment's example, the cell of clue 1 has remaining neighbors A and C (D has been reduced to be safe). The cell with clue 2 has remaining neighbors A and C as well, and since B is a mine, our clue value is reduced to 1. Therefore the two equations for cells 1 and 2 are the same so we reduced the redundant equation to just A+C=1. The clue value for 3 is reduced to 2 because B is a mine, so the resulting equation is C+E+F=2. The full linear equations are then generated into an augmented matrix, where active variables in the equation are represented by 1s and inactives by 0. As explained in the approximation section of our analysis, if there are any columns of our matrix that consist of only zeros, we exclude those columns from our matrix to prevent our program from analyzing cells that are already declared as safe/mine or are inactive unknowns. For our example columns 2 and 4 are removed to create a simpler matrix. The final reduced matrix is now in the form we need, with A+C=1 and C+E+F=2, without any extraneous variables.

```
For the system of equations:
A + 0B + C + 0D + 0E + 0F = 1
0A + 0B + C + 0D + E + F = 2

We represent this by:
[1 0 1 0 0 0] = [1]
[0 0 1 0 1 1] = [2]

And it can first be reduced to:
[1, 1, 0, 0] = [1]
[0, 1, 1, 1] = [2]

Since we had a zero column for the 2nd and 4th variables (i.e. B and D)
```

Figure 3

1.4.2 Generating Probability

For our six variable example above, if we were to include the inactive unknowns (zero columns), we would end up with 2^6 (64) potential vector solutions that we would need to check. However, if we eliminate these inactive unknowns from our equations, then we will reduce this total amount. Since B and D give us no helpful information (we already deduced what we could from them), we can safely look at only the remaining 4 variables A,C, E and F. This then leads to only 2^4 (16) potential solutions that we need to check. The generated solutions can be seen below.

```
The possible solutions for the reduced matrix are:
[[0 0 0 0]
 [0 0 0 1]
 [0 0 1 0]
 [0 0 1 1]
 [0 1 0 0]
 [0 1 0 1]
 [0 1 1 0]
 [0 1 1 1]
 [1 0 0 0]
 [1 0 0 1]
 [1 0 1 0]
 [1 0 1 1]
 [1 1 0 0]
 [1 1 0 1]
 [1 1 1 0]
 [1 1 1 1]]
```

Figure 4

We check if these solutions are valid by taking the dot product between them and our matrix. If the resulting vector is equal to our clue vector (in our case it would be [1, 2]), then we have a valid solution. Of the 16 possible vectors, only 3 yielded valid solutions to our system of equations as seen below. The final step would be to take the average value of each column (thus each variable), and that average value would tell us how likely that spot is to be a mine. As expected, we find that A has probability $\frac{1}{3}$ to be a mine, and the rest have probability $\frac{2}{3}$ to be a mine.

```
Now take the dot product of each potential solution with the system of equations
We find that the valid solutions are:
[0 1 0 1]
[0 1 1 0]
[1 0 1 1]

And the corresponding probabilities are:
[0.3333333333333333, 0.6666666666666666, 0.6666666666666666, 0.6666666666666666]
```

Figure 5

2 Basic Cost Agent

2.1 Data Analysis

The above plot shows the results of % Average Cost vs. Mine Density for our agent from Project 2 (in blue) and our agent that minimizes cost (in red). In general, as we increase the mine density of a board, we expect to see the percentage of mines stepped on increase (i.e. the cost increase) for any agent. Both agents can solve the game with basic logic when the mine density is low around 0.0 - 0.2 and both struggle to obtain enough information when the density is higher around 0.8 - 1.0. Therefore, we expect each of their performances to be similar in these regions and this is exactly seen in Figure 6. When the mine density is within the range of 0.2 - 0.8, we see that our cost agent ends up stepping on approximately 5% fewer mines than our agent from the previous project. Instead of picking randomly when we are stuck with no obvious moves, which is exactly what the Project 2 agent does, we use our simulated probabilities and make a decision based on these. Since we are now

accurately predicting what the probabilities of each active unknown cell is, we end up with more information about the board than if we just pick randomly. This results in a better chance to select a safe square than if we were to pick randomly.

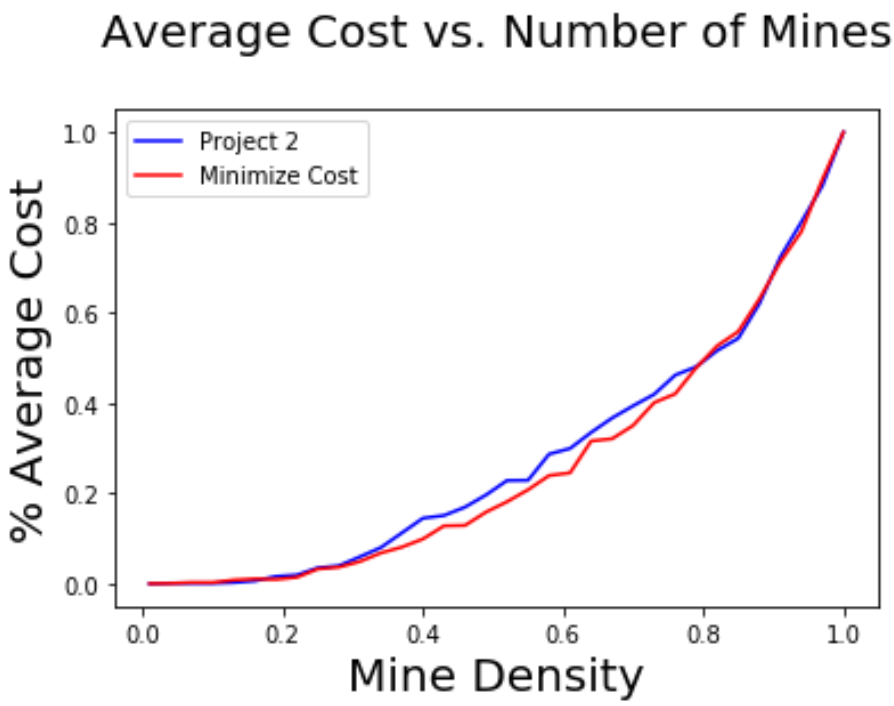

Figure 6

2.2 When and What to Pick

The methodology behind the cost agent was fairly simple, looking only at the cell with the lowest probability of being a mine and making our decision based on its probability value. In general, the probability that a completely random (i.e. inactive unknown) square is a mine is exactly $\frac{1}{2}$. Since we have no information about how many mines are left and we know the only possibilities are either mine or safe, then the $\frac{1}{2}$ probability is the only possibility for a completely random move. By utilizing this information, if the lowest probability for an active known square being a mine is less than $\frac{1}{2}$, the obvious choice for our cost agent is to pick that square since we have less of a chance of stepping on a mine. If the lowest probability is greater than $\frac{1}{2}$, then again, the obvious choice is to pick randomly from the inactive unknowns, since our chance of stepping on a mine is smaller. In the event that we have a tie, that is the lowest probability that an active unknown is a mine and the probability an inactive unknown is a mine, are both $\frac{1}{2}$, then we must break the tie. We decided that it would be best to reveal the active unknown square, as we would end up revealing more information about the board. Any information revealed around an active unknown square would potentially lead to more information about other surrounding active unknown squares and we would potentially be able to open up more relevant information for the game. Choosing an inactive unknown would only add the list of active unknowns and not progress our game in a helpful way.

3 Basic Risk Agent

3.1 Data Analysis

The agent for minimizing risk shows an improvement for a mine density greater than 0.2 and up until 0.8, which is similar to what we saw in the cost agent. In this region, the minimizing risk agent tends to make definite moves about 5% more often than the Project 2 agent does. This occurs because this agent tries to minimize the number of uncertain moves (risk) made in the game by utilizing information it gains from simulating moves and looking ahead.

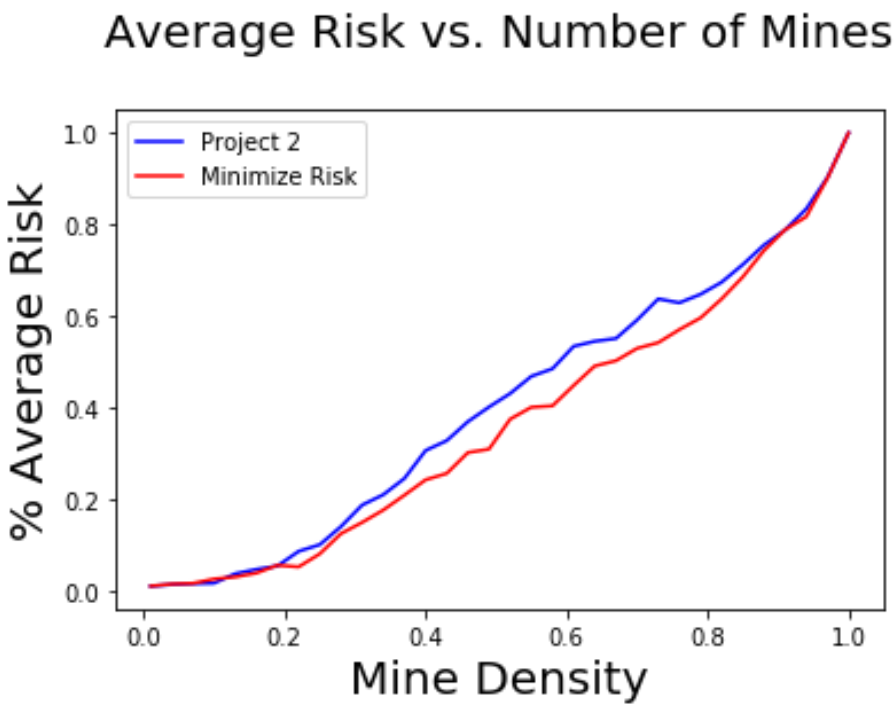

Figure 7

3.2 Methodology

Unlike the cost agent, which simply looks at the lowest probability of an active unknown being a mine, the risk agent has to look a step ahead. The risk agent simulates the outcome of an active unknown by marking it first as a mine and then as a safe, determining the resulting number of definite moves we uncover in each scenario. This was achieved by creating a temporary knowledge base with the current active unknowns and checking the possible neighbors of these cells. The agent then followed the basic rules set forth in the previous project to determine any obvious safe or mine cells and kept track of them in the temporary knowledge base. The results get stored into the variables R and S, where R is the number of squares the agent could work out if the possible move were a mine and S is the number of squares the agent could work out if the possible move were safe. We then insert both values into Equation 1 to find the expected number of squares that can be worked out when opening the current possible move.

$$qR + (1 - q)S \tag{1}$$

Where q is the probability of the possible move being a mine. This equation then tells

us the expected number of moves we will be able to immediately determine for a given cell. After simulating this value for each of our active unknowns, we then choose the cell that has the highest expected value to be revealed. The calculation of risk and simulation of sample possible moves is only necessary if nothing can be determined by the knowledge base after inferences are made, the safe and mine fringes have been traversed and emptied and moves that minimize cost are made. The "improved" decision making is in place of making a random move, since it chooses a cell with a greater probability of the rest of the board being solved, taking into account whether or not the cell could be a mine.

4 Bonus

We had expected that each agent would be better at minimizing its respective quantity, as that was the main priority for each of them. Surprisingly, we found that the risk minimizing agent is better when it comes to minimizing both cost and risk. Even though there isn't a significant difference between the two, the risk agent outperforms the cost agent by a very small percentage when minimizing cost. In hindsight, we think this result makes sense because the risk agent looks a step ahead in the game, whereas the cost agent only uses information from the immediate state of the board. Additionally, the minimizing risk agent works to minimize the number of uncertain moves and perhaps hits less mines overall for this reason.

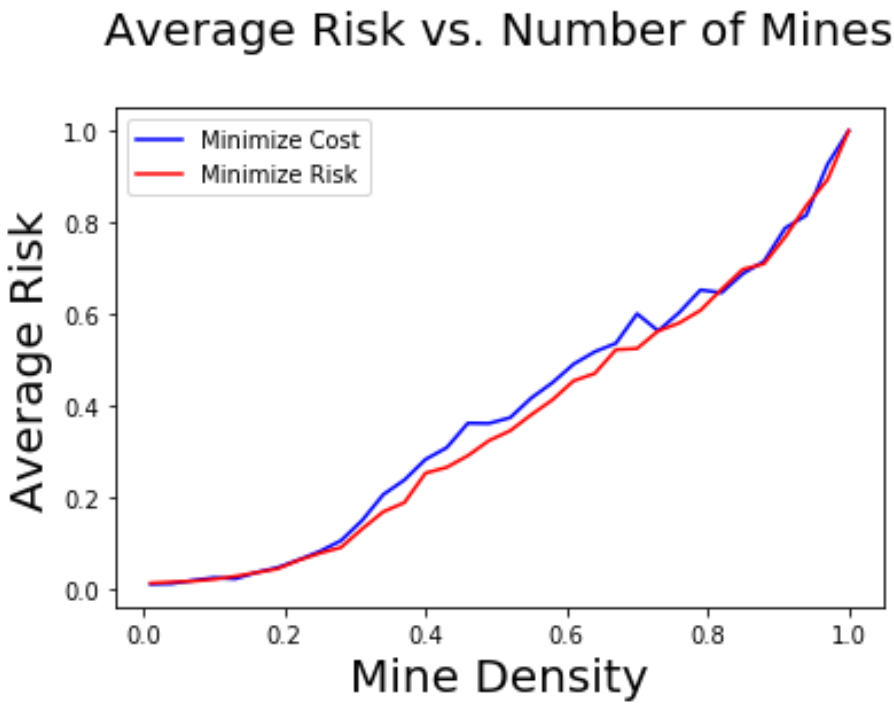

Figure 8

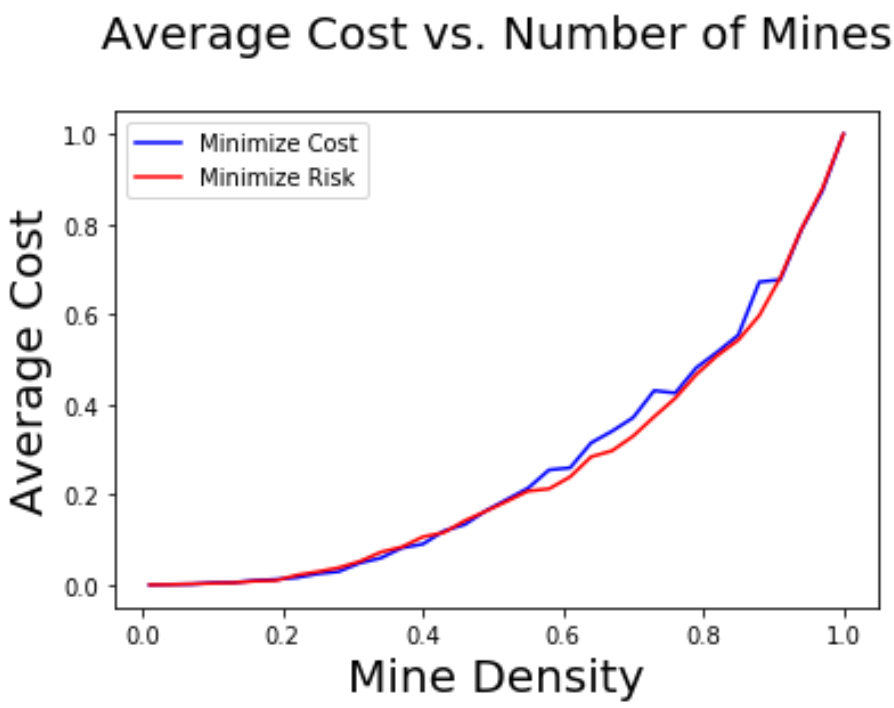

Figure 9

5 Improved Cost Agent

5.1 Basic/Improved Cost

For our improved agent we decided to build upon the basic cost agent that we previously created. As explained before, our basic cost agent currently assigns each active unknown a probability that it is a mine based on information we gathered from our inference step. It will then analyze the cell with the lowest probability and make a decision based off of the probability value. Our improved cost agent takes this same logic a step further by utilizing the concept of conditional probability in order to look a step further into the game. Therefore, when faced with similar scenarios, the two agents will tend to behave differently and make different decisions, even if the overall outcome is the same.

5.2 Simulation

In order to simulate conditional probabilities, we first needed to analyze all of the valid solutions that were previously generated. For example, if our system of linear equations contained variables [A,B,C] we would simulate clicking on A and find the probabilities that B and C are mines given A. We would continue this process for ${}^{n}P_{2}$ permutations, where n is the total number of variables in our equation, thus calculating all conditional probabilities are calculated for each variable. If we were to calculate the probability of B being a mine given A, we would use the following equation:

$$P(B\ mine|A) = P(A\ mine) * P(B\ mine|A\ mine) + P(A\ safe) * P(B\ mine|A\ safe) \quad (2)$$

With the list of valid solutions we have generated we can find the probability of A being a mine and the probability of A being safe. We then look for solutions where A

is strictly a mine— from those solutions we determine the probability of B being a mine. Solutions where A is strictly a safe are found and we determine the probability of B being a mine based on those solutions. All four of these probabilities are combined to find the total conditional probability for each variable being a mine given each of the remaining variables. Each variable will have n -1 possible conditional probabilities associated with it, and from this point we select the smallest probabilities from this set. Once the smallest probability has been selected for each variable we compare probabilities across our variables and select the variable that is associated with the smallest conditional probability— this is the next cell to be uncovered. However, if there is more than one variable that shares the smallest conditional probability we have to resolve the tie. We do this by replacing those tied conditional probabilities with their original probabilities (i.e. the probabilities that would have been used by our basic cost agent).

5.3 Comparison

Figure 10 shows that our improved cost performs slightly better than our basic cost agent. This occurs when the density is within 0.4 and 0.9, because outside of this region both methods can solve the game with basic strategies. Even though both agents attempt to minimize the cost of the next move, the advanced agent ultimately outperforms the basic one. By looking ahead and utilizing Equation 2 to find conditional probabilities of moves, the advanced agent ultimately steps on fewer mines.

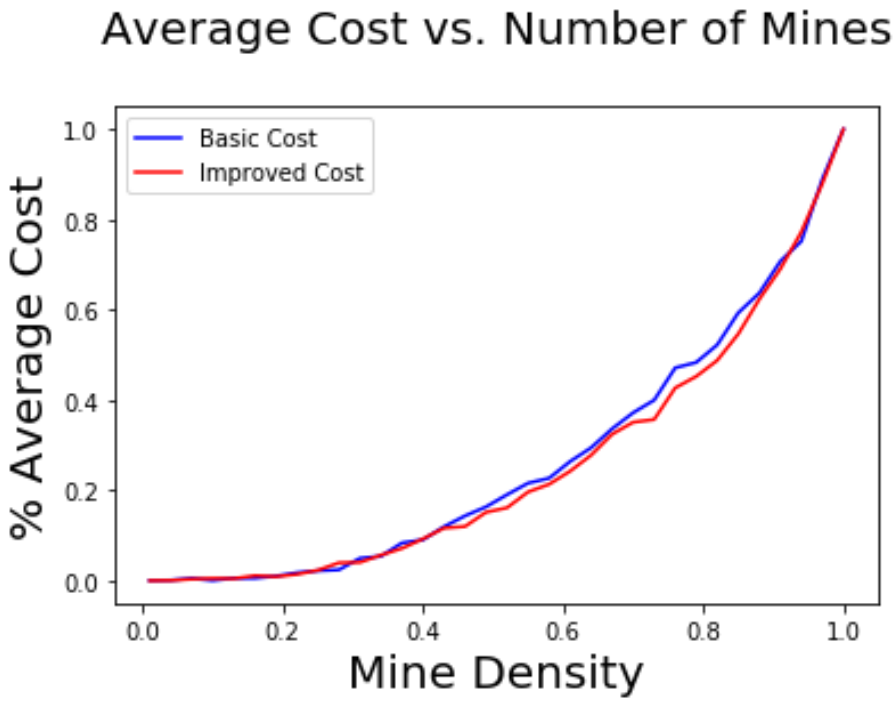

Figure 10

5.4 Things to do Differently

We could definitely make our improved cost agent better. Even though it performs better than the basic cost agent, it only simulates one step into the future. One of the biggest issues we had while doing this project was handling the vast amount of potential solutions that would occasionally arise for some boards. As we explained in Question 1, there were

some areas where we could simplify and make approximations to speed up our calculations. This ended up working well for the basic cost and risk agents, however, when we moved to the advanced cost agent similar problems arose. On top of the large amount of immediate probabilities we needed to simulate for each active unknown, we then also had to calculate every permutation of conditional probabilities. Since we didn't make any approximations for calculating conditional probabilities, we were only able to simulate one step into the future.

Member Contributions: All members contributed equally