SOLVING MINESWEEPER PUZZLE

ANALYSIS OF TWO METHODS

Shneka Muthu Kumara Swamy (muthu013)
Aaron Sattler(sattl024)
& Feng Liu(liux2824)

ABSTRACT

We compared various methods for solving the minesweeper problem.We considered a 9 X 9 Minesweeper Grid and divided into many parts and analyzed the results.This was mainly done to understand the efficiency of solving a minesweeper problem with multiagents . We have also done a detailed analysis about various difficulties in solving the problem using Reinforcement Learning and due to the complexity of the problem we have included ideas that may help in future analysis of these types of problems.

INTRODUCTION

Minesweeper is a single player puzzle which is a part of Microsoft Games.The Puzzle can be played and mastered quite easily. The puzzle essentially consist of M X N grid which has a mine or some information about the mine. Initially all these grids are not revealed and the status is not known to the player. The action of choosing the square may reveal a mine or can give some information about the position of the mine. The main aim of this problem is to reveal all the information squares without revealing any mine squares. A given grid contains P mines. The complexity of the problem increases with the increase in the number of mines.

Any player just playing some random moves has a chance of 3% to complete the game. The first choice that is taken is always taken at random and the probability that it might be a mine should also be considered. However, in some minesweeper games the mine placement is done after the first move to give every player a fair chance of winning. The probability factor in the game influences the winning, in which case the information is ambiguous and the player is often forced to guess, also the number of states that can be taken for every square is 9 thus increasing the complexity. This has made minesweeper a NP Complete Problem.

Given the complexity of the problem and the number of states involved it is very difficult to solve the problem with Reinforcement Learning and thus we have analysed the difficulties that

arise while solving the problem using this method. Another approach that we have used in this project is to solve the problem for small grid and then extend it further for the entire grid. This method gave an efficiency of about 50%. This turned out to be an unexpected result because intuitively solving the problem this way should give much better results.This made us probe more into the topic which helped us to gain a deeper insight about the problem.

PROBLEM

Consider a M X N matrix and this matrix consist of P mines. When any index in the matrix except the mines index is taken it gives information about the number of mines surrounding it. Using this information we need to locate the mines. These mines once found can be marked.

In our project we considered a 9 X 9 grid and the grid consist of 10 mines. We also assumed that the first chosen square by player will never be a mine and that a box chosen will reveal at least 5 squares surrounding it. This is because a minimum of 5 squares is required to solve the problem and the player can only take random moves till that point. The player always starts from the top left corner.

MULTI AGENT SOLUTION:

The multiagent minesweeper solution can be described in the following steps :

1. Division of the grids
2. Order of Evaluation
3. Results comparison and Analysis

DIVISION OF GRIDS :

To make multiple agents work on a single grid we must divide the area into many smaller regions so that each agent can work on a specific region. The grid partition in the project is done in 3 ways.

1. **USING A 4 X 4 GRID**:

X0	X1	X2	X3
X4	X5	X6	X7
X8	X9	X10	X11
X12	X13	X14	X15

These are used in case of the grids present in the corners and we solve for X0 , X1 , X2 , X4 , X5 , X6 , X8 , X9, X10.

After finding the values for the 9 squares highlighted the grid is moved by one column. This means that 6 values are already known and we have to solve only for the remaining 3 squares and this is moved to the maximum extent possible.

This grid is used for both horizontal and vertical traversals.

2. **USING 3 X 3 GRID**:

X1	X2	X3
X4	X5	X6
X7	X8	X9

As can be inferred from the box shown above (in 1 -- 4X4 Grid) , the last column in the problem cannot be solved and hence given the six values we solve for the last column of the problem.

3. **USING A 5X5 GRID**:

X1	X2	X3	X4	X5
X6	X7	X8	X9	X10
X11	X12	X13	X14	X15
X16	X17	X18	X19	X20
X21	X22	X23	X24	X25

This grid is used to solve the centre most region of the problem. Since the information required to solve is more in this grid when compared to any other grids considered above, usually all the grids mentioned above are solved first and this is kept towards the end.

ORDER OF EVALUATION:

The given grid with the required information can be solved in many ways. The most common way is to represent it in a form of matrix and then solve it deterministically. But a deterministic solution need not always be possible in which case we can choose something at random. We can also determine the position of the mine using posterior probability distribution.

We are using deterministic learning at first and then in conditions where these are not possible we are using posterior probability. The question as to why we should not use it the other way round, that is, doing posterior and then going for deterministic is attributed to the fact that in probability there is always a doubt as to what the state might be but this is not the case for deterministic solution.

The order of doing the execution that is deterministic followed by posterior was decided based on the results of running experiments on each method separately. It is seen that deterministic solution gave about 70% efficiency whereas posterior showed less that 50% efficiency.

IMPLEMENTATION :

The multiple agent system implemented uses shared memory, that is all the agents make changes to the same matrix . Shared memory is used because it decreases the complexity of communication between agents.The agents in the program are made to work sequentially however the grid method can also be used for parallel execution. There are some constraints to be imposed in case of parallel executions, like controlling the order of traversal of the grid.

The execution starts from the top most corner and the path to complete the outer square is taken and then the inner squares are executed. The program automatically quits if a mine is chosen and hence marking of all flags would mean proper completion. This is used to calculate the efficiency of the method.

1. **USING SYSTEM OF LINEAR EQUATION:**
   Consider the following grid :

X1	X2	X3
X4	X5	X6
X7	X8	X9

In this case a system of linear equations can be formed as follows,

eg) X1 = X2 + X4 + X5

X5 = X1 +X2+X3+X4+X6+X7+X8

and the known values can be used to solve the linear equation

In this method the value of any information is replaced as '0' and mine is replaced as '1' and it is subjected to the constraint that the output can be either 0 or 1.When 1 is got as the output then it means there is surely a mine and hence it is flagged. When the result got is 0 then there is some information which can be revealed.

We implemented this as a function in which the relation between the variables are stored in a matrix. For example let us consider the 4X4 matrix the representation is a 9 X 16 in which 9 represents the number of values we are solving for and 16 the information that will help in finding the solution. Due to the constraints imposed and the limited information available the method may not always give a solution.
(eg) : Consider the grid :

0	1	X3
0	1	X6
0	1	X9

This cannot be solved using linear equations method.
So for all the variables for which we cannot find a solution we use posterior probability method.

POSTERIOR PROBABILITY:

The probability of finding a mine in a given square given the information is called the posterior probability. Let us consider the following grid ,

0	1	X3
0	1	X6
0	1	X9

Posterior Probability is given by,

$$\text{Probability} = \frac{\text{Number given as information}}{\text{Number of variables surrounding it}}$$

Thus, the probability of X3 = 1/ 2 = 0.5

Once the probability is calculated then the box is opened or flagged. But the problem in this method is that if the probability is 0.75 then it means that in 100 trials 25 of the time we will choose the wrong result. This in turn decreases the operation efficiency.

EFFICIENCY AND COMPARISON

It was found out from the literature survey that the efficiency of solving the entire grid using the following methods is

1. Linear Equation Method = 90 %
2. Posterior Probability Method < 50%

By dividing the grids to enable multi agent operation and using both the methods in conjunction the efficiency was found to be < 50%

The main reason for the decline in efficiency percentage despite using the methods is lack of enough knowledge about the surrounding while dividing the entire region into smaller grids.

RESULT ANALYSIS:

1. There are many problems that can be created while making the agent work in a partially known environment.
   e.g., Consider the following grid:

0	0	0	0
2	2	2	X7
M	M	M	X11
X12	X13	X14	X15

The output given by the agent with just the given information is as shown above. The agent by just using the probabilities jumps to the conclusion that X10 is a mine but the solution got is inconsistent. This is one situations agents might get struck at often.

2. Consider an another major problem, when we first considered a grid we said that we need at least 5 squares to get a proper rational solution as an output and when we moved that grid by one column we assumed that we will get information about 6 squares but this is not always true.

(eg)

Info	Info	Info	X3
Flag	Flag	Flag	X7
M	M	M	X11
X12	X13	X14	X15

In this case since flag does not give any info about the surrounding environment we are basically left with no choice but to take a random decision.

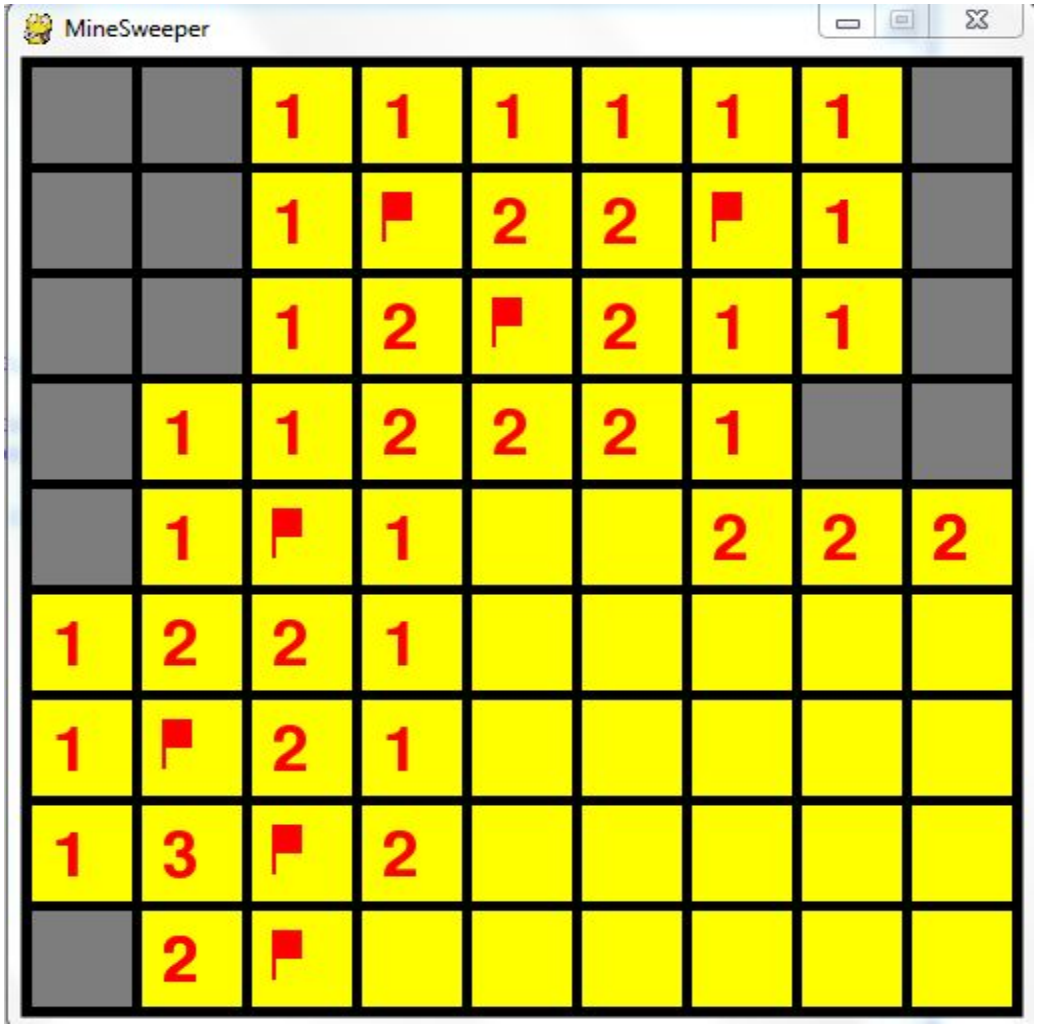
These are some of the reasons for not being to improve efficiency of Minesweeper problem.The actual tradeoff here is speed and efficiency. If the system performs fast the performance is not guaranteed.

IMPROVEMENTS SUGGESTED:

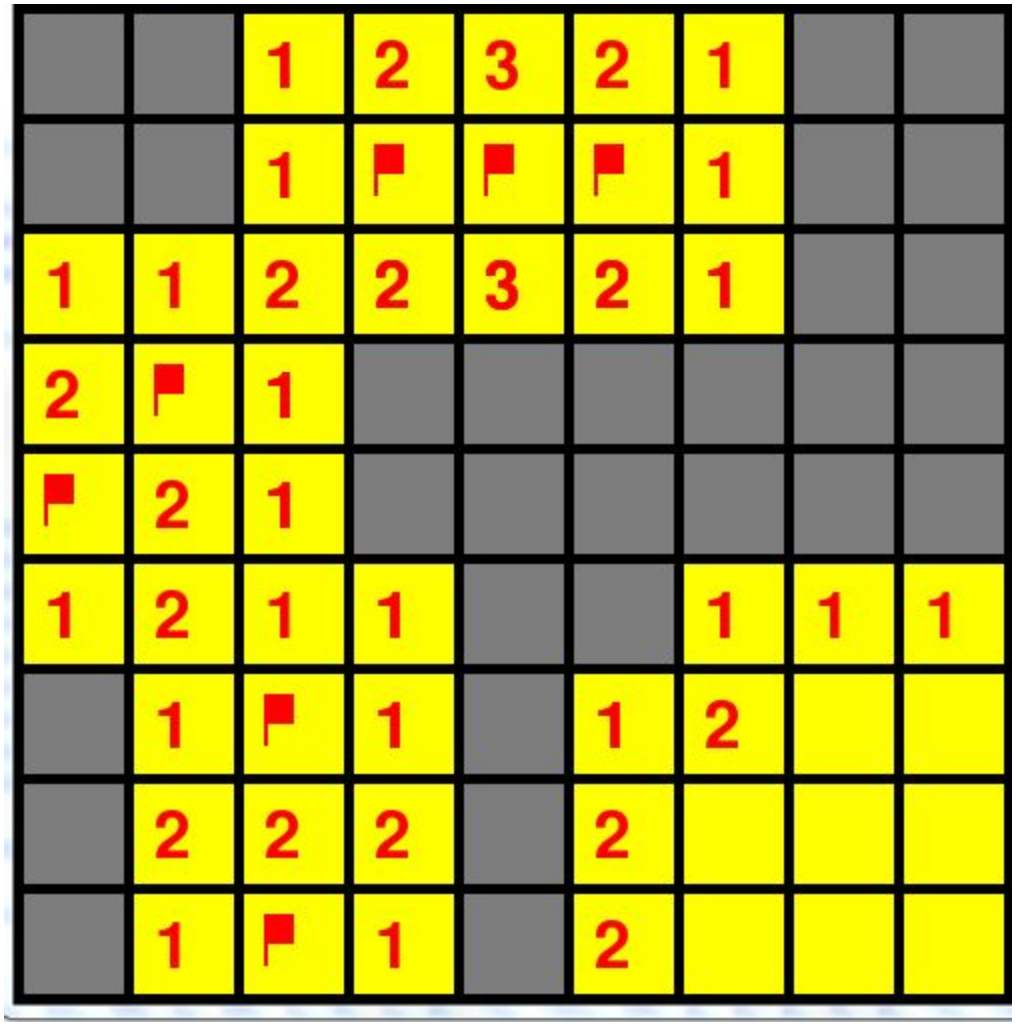
1. Now all the agents in the board are made to solve the grid once the encountered it instead of doing this we can make the agent to solve all other smaller boxes and then find the solution.
2. The size of the grid must be more for example if the problem we are solving is a 30 X 30 grid then taking a 9 X 9 smaller grid makes sense.

RESULTS :

1. Solving one horizontal and vertical region : (3 X 3) matrix



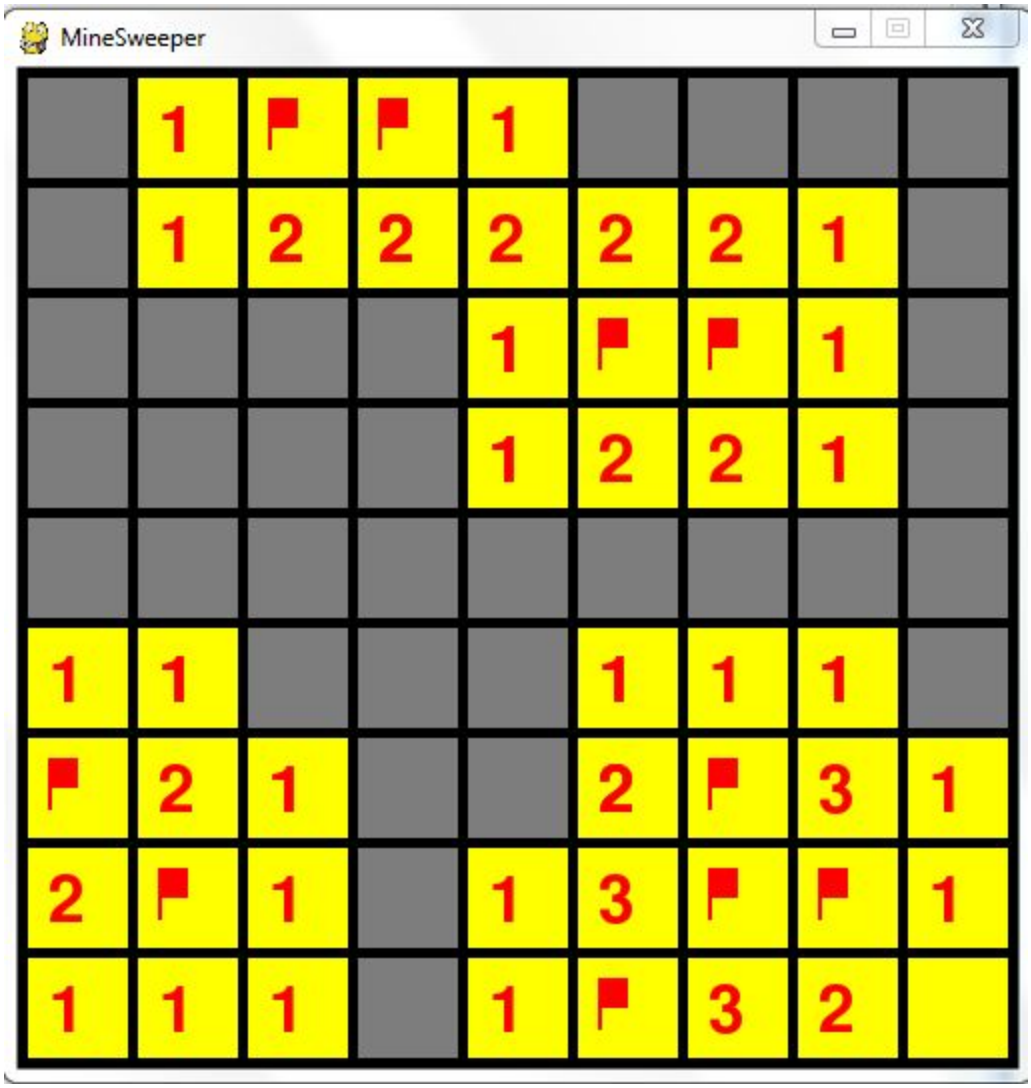
2. MINESWEEPER INTERMEDIATE SOLUTION (WITHOUT 5 X 5)



The above picture is taken before using one horizontal traversal that is for the bottom most rows and before using the 5 X 5 grid and it can be inferred that in most of the cases the traversal using the 5X5 grid may is not required.

Using this information would decrease the number of agents required to find the solution this will also decrease the time of operation.

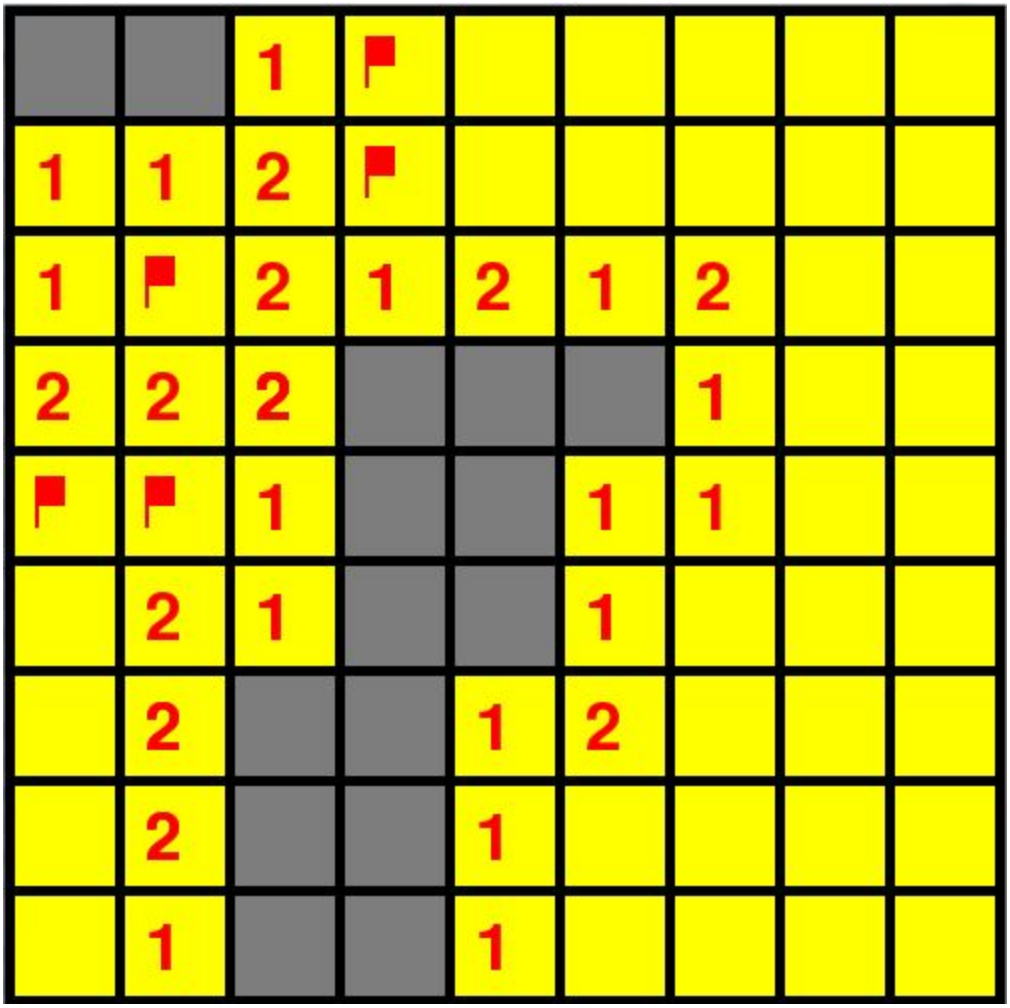
3. MINESWEEPER FINAL SOLUTION:



As it can be seen in the above result the agent is said to have completed the task if it has flagged all the mines properly.

The last information square left unopened need not be opened because a solution is already found.

4. PARALLEL WORKING AGENTS (INITIAL ATTEMPT) -- THREADING



The result shown above is an attempt to make the agents work in parallel. This is done for only two traversals and it is implemented using Threads.

REINFORCEMENT LEARNING

Reinforcement learning for the minesweeper game is what we planed to implement and use as the competitive approach to compare with the "linear equation + probability" approach. Unfortunately, we later found out it is a such a difficult task and we fail to accomplish it.

How do we plan to use reinforcement learning on minesweeper?

Our idea is that, when the deterministic approach cannot give us a certain more, we can use apply the policy by reinforcement learning. In our project, we view the minesweeper game as an MDP model.

We design our model as the following:

"States": one state is one valid configuration of the grid. One configuration of the grid contains the information that every square could be either unknown (with the value -1) or known (with the value 0-8, which represents the number of mines surrounding).

"Actions": In each step, we probe one unknown square and it will lead us to a new state.

"Reward". In a simpler model, we could just define the state rewards as--- if the action leads to a failure state, set the state reward to be -1; otherwise set it to 0. However, we could also define it in a more sophisticated way. For the configuration with more possible arrangements for the mines, we give it a less positive value; for the configuration with less possible arrangements for the mines, we give it a larger positive value. In this way, it should be expected that we will have a higher chance to meet with more deterministic states.

Why doesn't it work?

The model seems to be natural at first glance. But when we look deep into it, we found out that it is actually not applicable or useful. The main problem is that the state space is too large. For an a×b grid, without considering the validation of the configuration, there could be as many as $10^{(a-2)(b-2)}\times 7^{(2a+2b-4)}\times 5^4$. Of course, most configurations are invalid in this case, because they either contradict to themselves on the known square or contradict to the global constraint (the total number of mines). We know for a 4×4 grid, there are about 3×10^6 total valid configurations (states) [3]. However, it is still two large for the naive reinforcement learning such as Q learning to solve.

We first thought we could reduce the state space significantly by exclude the states that can be solved deterministically. But it didn't work. The computation is still beyond what we can deal with and it seems there are no obvious ways to significantly reduce that number. And it is extremely difficult to find out all the states that can be solved deterministically.

In fact, in Nakov and Wei [3]'s endeavor to apply a reinforcement learning strategy on the minesweeper game, they can only success at a 4×4 board, with numerous techniques for cutting the state space such as working on the problem graph, exploiting symmetries and sure moves. And they cannot go beyond 4×4.

Endeavours to solve the problem

We've tried different ways to solve the problem.

The first thought is to define a model that has less states, such that the reinforcement learning will not encounter computation explosion. Unfortunately we cannot. There is another natural graph view of the minesweeper as a POMDP model[4], but it doesn't help because the state space is as large. And we cannot find other models that can be described using less parameters.

Now let's go back to the MDP model. Notice that the state space actually can be represented by a 81 dimensions vector, in which every dimension denotes the value of one square. If we can reduce that dimension, we can reduce the states we need to deal with significantly.

We then had an idea to divide the grid into small grids and apply reinforcement learning on the small grids. Using every small grid as one dimension and find a way to define the value of every small grid, we then can define the reinforcement learning. The problem is, whether there is a way to define the values for the small grid is beyond our knowledge, also the definition of reward function is a difficult one.

Then how about directly using dimension reduction methods on the original 81 dimension vector, such as Linear discriminant analysis. In this way, we don't need to define a new reward function, because it is a one-on-one mapping. There are a lot of more sophisticated dimension reduction methods, which can be our candidates. But still, whether using dimension reduction this way on reinforcement learning is applicable or not is beyond our knowledge.

At last, we turn to other methods of solving reinforcement learning with large state space. There are a lot of potential approaches, such as policy search, function approximation and so on. In fact, a great Backgammon solver is implemented using a TD(X) reinforcement learning algorithm, based on neutral network, with 40/80 hidden units[8], which makes us believe that a reinforcement learning on minesweeper is possible.

CONCLUSION :

We explored various methods for solving minesweeper. Instead of the conventional way of solving the entire grid we wanted to try someway that would help use multiagent and hence we worked with the above mentioned procedure and found the solution to be less than 50% effective.

This made us look for more solution methods and hence we did a detailed analysis on Reinforcement learning hoping that we could make the agent learn its own strategy but we came to a conclusion that implementing Reinforcement Learning for the agent is very difficult, although possible.

TEAM MEMBER'S CONTRIBUTION:
Shneka :Programming for Minesweeper Grid and Multiple Agent
Aaron : Reinforcement Learning and Literature Survey
Feng : Reinforcement Learning and related discussions

Works Cited

[1] L. P. Castillo and S. W. Fraunhofer, "Learning Minesweeper with Multirelational Learning". *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, August 9-15, 2003.

[2] R. Kaye, "Minesweeper is NP-complete", *Mathematical Intelligencer*, vol 22, number 2, pp 9-15, 2000.

[3] Nakov, P. and Wei, Z., "MINESWEEPER, #MINESWEEPER", University of California at Berkeley. May 14, 2003

[4] "MineSweeper: Where to Probe?" *Research Report* RR-8041, 2012, pp.26

[5] "Value Function Approximation in Reinforcement Learning using the Fourier Basis", *Technical Report* UM-CS-2008-19

[6]“Dimensional reduction for reward-based learning”, *Network: Computation in Neural Systems* September 2006; 17(3): 235–252

[7]“Minesweeper on graphs”, *Applied Mathematics and Computation ,*Volume 217, Issue 14, 15 March 2011, Pages 6616–6623

[8]“TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play”, *Neural Computation* Volume 6 Issue 2, March 1994 Pages 215-219