# CS181 Artificial Intelligence I Project: A Minesweeper Using Inference, CSP and CNN

**Chenyang Zhang, Chunxu Guo, Jiahao Huang, Qianyu Liu, Tianyi Zhang**
ShanghaiTech University
zhangchy1@shanghaitech.edu.cn guochx@shanghaitech.edu.cn
huangjh@shanghaitech.edu.cn liuqy@shanghaitech.edu.cn
zhangty2@shanghaitech.edu.cn

## Abstract

*Minesweeper is a single-player puzzle computer video game which is proved to be Turning Complete and in a class of mathematically difficult problems known as co-NP-complete as well. We propose 3 ways containing logic reasoning, CSP backtracking, CNN and their combinations to solve this question. By binding deterministic with non-deterministic methods we finally achieve a SOTA outcome.*

## 1. Introduction

Minesweeper is a single-player puzzle computer video game written by Robert Donner ami Curt Johnson which include in Microsoft Windows© in 1991, whose goal is to clear a rectangular board containing hidden 'mines' or bombs without detonating any of them. This game originates from the 1960s and has derived different variants on many computing platforms. Minesweeper is proved to be Turning Complete and in a class of mathematically difficult problems known as co-NP-complete as well, so exploring algorithms to solve the Minesweeper game may inspire us to work out other related problems.

Minesweeper game is defined in a custom-sized square grid board with a certain number of mines. The most classical and standard size of mine board is $9\times9$ (beginner), $16\times16$ (intermediate), $16\times30$ (advanced), and the standard mine density of advanced one is 20.63%. (Usually mine density will be higher than this ratio.)

There are 2 main competitive criteria for Minesweeper players, one is speed and the other one is correctness. The model in this report tends to reveal safe cells as much as possible in an acceptable time limit. Besides, the Minesweeper game might sometimes become a non-deterministic problem, which is roughly regarded as a classical probability model, supposing that all the mine matrices are randomly generated in the beginning, satisfying the uniform distribution. Under this circumstance, neither people and computers could reveal a risk-free cell.

Playing the Minesweeper game requires logic, arithmetic, and probabilistic reasoning knowledge based on spatial relationships on the mine board. Up to now, multiple artificial Intelligent algorithms have been proposed to solve minesweeper game, such as Inference, Constraint Satisfaction Problem (CSP), Partially Observable Markov Decision Process (POMDP), SAT solver, neural network and so on. Considering their performance as well as related with our course, we design an algorithm model combining logic, CSP, and convolution neural network (CNN). The most complicated case among our test conditions is $16\times30$ with 20.83% mine ratio.

## 2. Related works

The studies of strategies of Minesweeper date back to the 1990s. Most researches can be categorized into two classes. The first one is combinations of constraint-based approaches and deterministic heuristic methods, such as cellular automation (Adamatzky 1997) [1], constraint satisfaction problem (Studholme, 2000) [5], SAT solver (Nakov and Wei, 2003) [4], and graphical algorithms (Kamenetsky and Teo 2007)[3]. These approaches can handle deterministic cases fast and perfectly, but slow or missing accuracy when dealing with non-deterministic cases. With the development of Reinforcement Learning and Deep Learning, some more methods appear, these learning methods are categorized as the second approach. Gardea, Koontz, and Silva came up with a Q-learning method [2] to solve Minesweeper but the winning rate is pretty low, for the state of the agent is hard to define. Ryanbaldini solves this problem using a CNN, and the winning rate is acceptable, however, it may fail in some simple deterministic cases since the neural network itself is a black box, whose prediction is difficult to explain.

## 3. Basic Ideas for Minesweeper

In a Minesweeper game, players will be given a certain sized grid board and the number M of unknown mines. Initially, all the sub-squares or the cells are blank, hidden among them are M mines distributed uniformly at random. To clarify the subsequent description, we raise some useful concepts here.

- **Neighbors**: unrevealed one in 8 surrounding cells are all the neighbors of a given cell.
- **Set**: A group of unmarked neighbors around a certain number cell.
- **Game Map**: Real-world interaction graph, a visual mine board with cells blank or filled with integer $n$. $n$ varies from 0 to 8.
- **Mine Map**: Ground truth, the real mine distribution that we do not know when playing.
- **Cell State**: At the beginning of the game, all cells are signed with an initial state as $\{1,0\}$, which means not sure about whether a cell has a mine. If a cell has been inferred as safe definitely, update the cell state as $\{0\}$. Else if a cell has been inferred with mine definitely, update the cell state as $\{1\}$. Otherwise, stick to the original cell state.
- **Transfered Map**: A local graph used in CSP phase, each cell in this graph are related to a cell state.
- **Turn**: Every time the player operates once is a turn.

At each turn, a real-world player can select three possible actions: to flag a cell as mine, to unmark a flagged cell, and to reveal a cell. But for AI robots, it would only maintain the transformed map corresponding to the game map and click on a safe cell with cell state = $\{0\}$, unless no such cell is safe. Under that uncertain circumstance, clicking depends on the probability for the cell to be a mine.

By clicking, the player will safely reward a number behind(sometimes a large area will also be revealed) or lose the game immediately after detonating a mine. Until all the safe cells are revealed, the player wins.

## 4. Methods

The main pipeline of our algorithm model in this report is shown below (Fig 1), involving junior and senior logic reasoning methods and probabilistic prediction.

### 4.1. Overview

Junior logic reasoning, the fastest and simplest cell assignment stage in our model, which is placed at the most front, only uses the adjacent $3\times3$ patches' constraints to figure out deterministic safe and mine-behind cells. Over half of the situations could be solved after this stage. While sometimes only considering adjacent information is not enough, a senior logic reasoning method, CSP, is followed. The magic power of CSP is embodied in cell constraint merge and division. In other words, it connects some constraints on non-fixed shaped patches, and advanced compare all the patch pairs instead of adjacent ones. The complexity of this method grows almost as exponential as the scale of the mine board due to the rapid increase of constraints space. Backtracking may help us pruning a lot but it is still much slower than simple inference. CSP is also used to figure out deterministic safe and mine-behind cells.

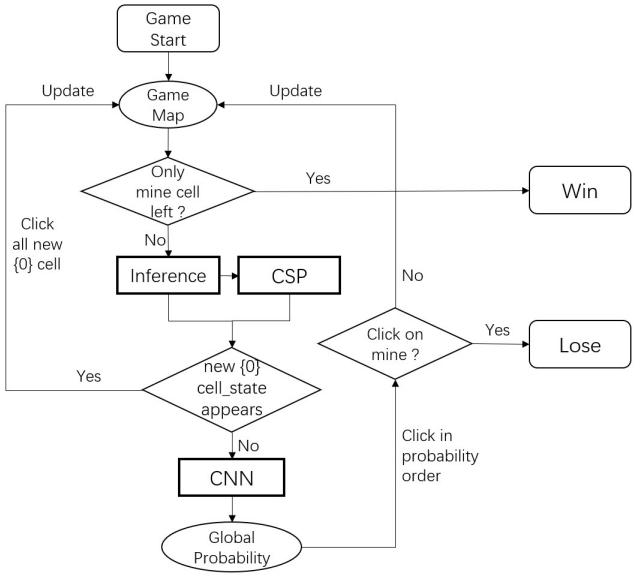


Figure 1: **Pipeline.** There are 3 stages in our pipeline: Logic inference, CSP search, and CNN prediction.

Either after the Inference or CSP stage, if AI robot could click on some safe cells, Game Map and Transferred Map will be maintained and reveal some new mine information which could be used for constraint update. Inference and CSP can run as far as possible with such regenerative feedback. Unavoidably, sometimes the situation will fall into a 'dead corner' and bring about a 'guess' turn. None of the remaining unrevealed cells can be assigned with a deterministic state, and the only choice left is to click on the most likely cells without mines. CNN, the last module of our workflow in Fig 1 helps us to build a global probability table. The fine grid kernel helps to work out neighbors' contribution, and the combination of multiple convolutional layers provides a greater field of reception, which enables us transform local constraint relationships to global probabilities.

### 4.2. Inference

Inference is the most basic step in our algorithm applying 4 basic rules as followed:

- **Theorem 1** : If #(items) in set is equal to number on cell minus mine neighbours,then all the cells in set can be marked as mines
- **Theorem 2** : For 2 number cells **a** and **b**, suppose **A** and **B** are their corresponding cell set, if **A** $\subseteq$ **B** and number on **a** minus #(marked neighbours) is equal to number on **b** minus #(marked neighbours), then cells in complement of **A** refer to **B** can be all revealed safely.
- **Theorem 3** : If #(marked neighbors) equals to the number on a cell, then you can reveal all the unmarked neighbors.
- **Theorem 4** : For 2 number cells **a** and **b**, suppose **A** and **B** are their corresponding cell set, if **A** $\subseteq$ **B** and number on **b** minus **a** equals to #(neighbours) of **b** minus #(neighbours) of **a**, then cells in complement of **A** refer to **B** can be marked as mines.

Figure 2: **Theorem 2**: According to the **Theorem 2**, we can reveal the cells marked safely.

Figure 3: **Theorem 4**: According to the **Theorem 4**, we can deduce the cell with mine.

Logic Inference enables us to find the confirmed results, which is the basic method to win the game. By applying the Logic Inference we can reach 70.7% winning rate in $9 \times 9$ Map with 10 Mines and 31.4% in $16 \times 16$ Map with 40 Mines as Table 5.1 shows.

### 4.3. Constraint Satisfaction Problem (CSP)

The CSP algorithm with backtracking used in minesweeper is a quite efficient algorithm. However, unlike traditional CSP problem that finding the solution that satisfy constrain is finished, in this algorithm, all possible solution to current constrain are needed to be found. The algorithm works as follows:

1. Each cell is a variable and each variable have 2 possible assignment: 1 or 0. In minesweeper, 0 means that this cell is not mine and it's safe to click and is a mine for 1.
2. Each variable is tested with value of 1 or 0. At each step of the assignment, all of the currents constraints are checked in case of a possible violation. If a constrain is found to be unsatisfied by one step of the assignment, backtrack and try another assignment. If this backtrack doesn't work, backtrack again, and try another assignment that 2 steps previous. Tally all constraints satisfied assignments.
3. If a variable is assigned only 1 value in all possible assignments, then it is a deterministic cell, we can infer that cell is mine or not. If this variable is assigned with more than 1 value, it is a non-deterministic cell and we can do nothing to it.

| | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|---|
| **0** | · | · | · | · | · | · | · | · |
| **1** | ? | ? | S | · | · | · | · | · |
| **2** | 1 | 2 | S | · | · | · | · | · |
| **3** | · | 1 | M | S | S | ? | · | · |
| **4** | · | 1 | 1 | 1 | 2 | ? | · | · |
| **5** | · | · | · | · | 2 | ? | · | · |
| **6** | · | · | · | 1 | 2 | ? | · | · |
| **7** | · | · | · | 1 | M | ? | · | · |

Figure 4: **Variables and Constraints.** Cells that labeled with S meaning safe, M meaning mined. In CSP algorithm one can infer some cells that are deterministic. In our algorithm we only infer edge cells.

At each step of minesweeper, we need to generate our constrain from the gaming board. Below we use mathematical language to describe the constrains that generated from board:

1. $x_{i,j}$ is either mined or safe:

$$\forall c = (i,j), x_{i,j} \in \{0,1\},$$

2. A cell $x$ with non-negative value is safe:

$$\forall c = (i,j), (K_{i,j} \geq 0) \Rightarrow (x_{i,j} = 0)\,,$$

3. the non-negative value on a cell c is the number of adjacent mined cells:

$$\forall c = (i,j), (K_{i,j} \geq 0) \Rightarrow \left( \sum_{c'=(i',j') \in \alpha(c)} x_{i',j'} = K_{i,j} \right).$$

### 4.4. Convolutional Neural Networks (CNN)

Having solved all deterministic case, the game state is sent to the CNN module. In this stage, the non-deterministic game state map is converted to a probability map which shows the probability for each cell to be a mine.

The reason why CNN can be used in Minesweeper fits with intuition. Firstly, $3 \times 3$ local area is of great importance in the Minesweeper game, for it contains all local information for a cell. Secondly, $3 \times 3$ kernels in convolutional layers can slide in the whole game state map, which means the kernel can manipulate similar local areas which locate in different areas in the whole map. Lastly, since combining many convolutional layers gives a larger receptive field, we may find more global information to predict the global probability map using CNN.

As Fig 7 shows, the net architecture is pretty simple, which consists of 5 convolutional layers with $3 \times 3$ kernels to extract features and 3 fully connected layers to map these features to a probability map. This makes the training process can be done in about 20 minutes, even using our laptop CPU.

The training process is pretty easy and not resource-consuming. We generate 6000 random game states, which are pre-clicked a few times, providing abundant information for convolutional layers. The input to the net is a pre-processed game state map of dimension $1 \times H \times W$, replacing all hidden cells with number $-1$. The predicted probability from the net is generally in the range of (0, 1), but some values may large than 1 slightly, without loss of representativeness. As Fig 5 shows, the net's prediction loss on validation set converges smoothly to a low level.

To target valid information, we focus on the 'edges' in the game state map. If a cell's neighbors are all hidden, this cell cannot get much information basically, thus we ignore the prediction error in this cell. For example, for the game state map in Fig 6, the edges are labeled with red boxes, which means they are at the edge of hidden cells, adjacent to the information (numbers).

## 5. Experiment and Result

### 5.1. Experiments

Winning rate is the most objective and persuasive test criteria for the Minesweeper game. We run our model for 1000 rounds of the real Minesweeper game, and count the winning rate in the 1000 games.

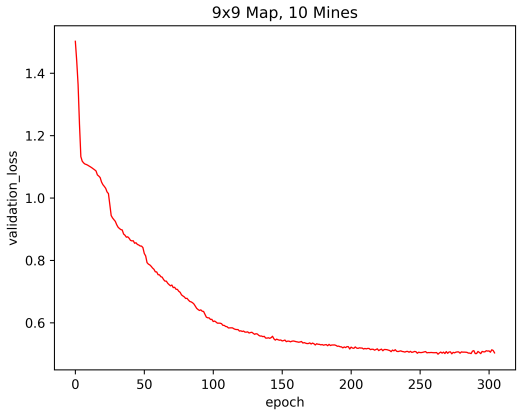


Figure 5: **Loss on validation set.** As training goes, the net's prediction loss on validation set converges smoothly to a low level.

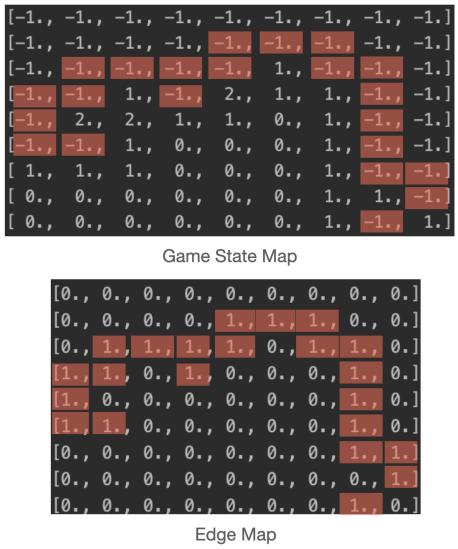


Figure 6: **Edges.** The edges are labeled with red boxes, which means they are at the edge of hidden cells, adjacent to the information (numbers).

### 5.2. Results and Analysis

As minesweeper is designed for human players and challenges mainly the speed of human reaction rather than inference ability, the common logic method achieves satisfying results, which is the formulas human players learn from the game.

The rules adopted in our logic method are a specific version of the minesweeper rule, which is the number of mines in neighbor cells should sum up to the figure of the center. Or in other words, the minesweeper rule is the generalized version of those logic rules, so we adopt CSP to search for

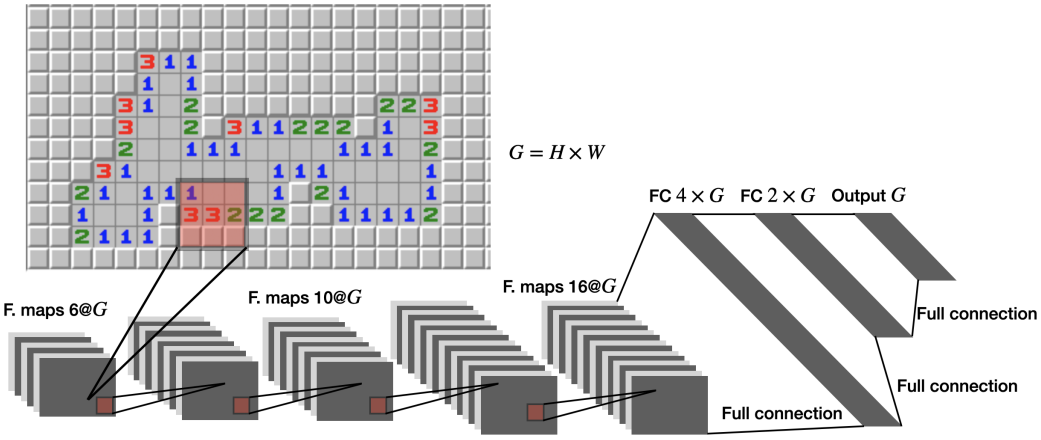


Figure 7: **Net architecture.** The net consists of 5 convolutional layers with $3 \times 3$ kernels and 3 fully connected layers to map the game state map to a probability map. Then the unrevealed cell with min probability will be clicked.

Table 1: **Winning rate in 1000 games with different strategy choices**

| Approach | $9 \times 9$ Map, 10 Mines | $16 \times 16$ Map, 40 Mines |
|---|---|---|
| Random Pick | 0.0% | 0.0% |
| Logic | 70.7% | 31.4% |
| CSP | 45.0% | 1.0% |
| Net | 32.8% | 0.0% |
| Logic + CSP | 84.5% | 48.7% |
| CSP + Net | 77.0% | 7.0% |
| Logic + CSP + Net | 95.0% | 66.1% |

the probable mine distribution, and it is supposed to solve every circumstance we are able to, despite the restriction of the search depth. As shown in the result, due to the restriction of the search depth, and larger global variable domain compared to the local one in the logic method, it shows a huge drop in the winning rate compared to the logic method.

So, the hybrid version of Logic Inference and CSP makes a big difference. We successfully combine the efficiency of the logic and the power of the CSP. Those simple cases common at the beginning of a game are mostly solved in the logic inference stage, and those rare cases are solved by CSP at the end. As we can see in the result, the winning rate of the hybrid version is higher than both Logic and CSP methods separately.

Known as a game with randomness, the minesweeper game often has multiple solutions for the current state, which is not able to be solved by inference. We train a CNN to solve this circumstance, using mine probability for each cell at a global level to predict which cell should be chosen in the next turn. Surprisingly, the winning rate improves from 32.8% to 77.0% in the beginner mode and from 0% to 7% in the intermediate mode, demonstrating the logic approach and probability approach are auxiliary to each other, and our Inference + Probability hybrid approach is pretty effective.

Lastly, we combine Logic Inference, CSP, and Net to our final hybrid version model. We reached a winning rate at 95.0% for the beginner mode and 66.1% for the intermediate mode.

## 6. Conclusion and Discussion

### 6.1. Conclusion

In this report, we've studied the Minesweeper game by designing an acceptable Algorithm model combining logic reasoning and probability to gain a higher rate of success with a given off-line mine map. Up to now, it is a meaningful trial taking advantage of every single method in this model.

### 6.2. Discussion

At the first stage of our workflow, we implement a simple logic inference module targeted at many obvious solutions considering all adjacent $3 \times 3$ patches in the half-revealed mine board. To excavate more hidden information existing among nonadjacent regions, we apply the CSP method. However, the state space is growing exponentially along with the difficulty level of the Minesweeper game increasing. Involuntarily some filtering methods such as backtracking and forward checking have been tried to speed up our CSP stage. It's worth noting that the success ratio of pure Logic method is higher than the success ratio of pure CSP method for merge and division step number limitation in CSP. Using more constraints relationship leads to a much longer time, thus CSP may not get more certain solutions

than simple logic does. That is the reason we lay out the Logic inference module at the first stage of the model we designed.

While no risk-free solutions could be proposed by logic Inference and CSP module, mine probability of board cells helps to sustain the game process. A convolutional neural network can use receptive fields of various ranges, converting local constraints to global probabilities through the full connection layer. During the experiment we notice that sometimes CNN will make stupid mistakes on several cells, therefore we cimbine the result of logic inference with CNN predicted action, in case it choose a mine that we have found.

### 6.3. Future Works

There are still several additional approaches that could explore related to our current research. One of the improvement key points is to cut down the state space in CSP module as much as possible, applying techniques like Generalized Arc Consistency and pruning could be feasible thoughts. Reinforcement Learning with deep Q-learning and extracting and weighting typical features for mine state are also an attemptable idea. Besides, transform the Minesweeper game into some different views may also inspire us new ideas to contribute to the game state, which could dramatically reduce the algorithm complexity.

Generally, though the rules of Minesweeper is very simple, it is an interesting and valuable game to explore. An efficient algorithm not merely helps to solve the Minesweeper game itself, but also contribute to conquer NP-hard problem.

## 7. External Resources

### 7.1. Libraries

- PyTorch: Training the network.
- NumPy: Calculations.
- matplotlib: Drawing the loss plot.

### 7.2. Minesweeper API

Our Minesweeper API is slightly modified from GitHub repo ryanbaldini/MineSweeperNeuralNet
`https://github.com/ryanbaldini/MineSweeperNeuralNet`.
And we have packed the API in our code, so you can run it directly.

## References


[1] Andrew Adamatzky. How cellular automaton plays minesweeper. *Applied mathematics and computation*, 85(2-3):127–137, 1997.

[2] Luis Gardea, Griffin Koontz, and Ryan Silva. Training a minesweeper solver.

[3] Dmitry Kamenetsky and Choon Hui Teo. Graphical models for minesweeper project report. 2007.

[4] Preslav Nakov and Zile Wei. Minesweeper,# minesweeper. *Unpublished Manuscript, Available at: http://www. minesweeper. info/articles/Minesweeper (Nakov, Wei). pdf*, 2003.

[5] Chris Studholme. Minesweeper as a constraint satisfaction problem. *Unpublished project report*, 2000.