

**Foundations of Artificial Intelligence**

***Solving Minesweeper with AI: A Hybrid Approach Using logic and probability***

***Team 4***

*Jones, John K*

*Muthukumarasamy, Shankar*

*Shu, Alex Banchao*

AI 801 (Foundations of Artificial Intelligence)

[Spring, 2025]

**Document Control**

**Work carried out by:**

| **Full Name** | **Email Address** | **Exhaustive list of Tasks** |
| --- | --- | --- |
| Jones, John K | jkj5774@psu.edu | Loops Update |
| Muthukumarasamy, Shankar | sxm6823@psu.edu | Algorithm Update |
| Shu, Alex Banchao | bxs5825@psu.edu | Fixing bugs |

**Revision Sheet**

| **Date** | **Revision Description** |
| --- | --- |
| 2/16/25 | Draft started |
| 2/25/25 | Added the challenges and objectives |
| 4/28/25 | Started Final Write Up |
| 4/30/25 | Added Experiment Result |
| 5/1/25 | Added Discussion |
|  |  |
|  |  |

**TABLE OF CONTENTS**

[1 Introduction 3](#_Toc944514453)

[2 Problem Statement 4](#_Toc271537187)

[3 Challenges 4](#_Toc1480982295)

[3.1 RELATED WORKS 4](#_Toc2018649835)

[3.2 IMPORTANCE AND IMPACTS 5](#_Toc1446130452)

[4 Data Collection and Preprocessing 6](#_Toc1830879850)

[5 Methodology 6](#_Toc6681267)

[6 Results and Interpretation 11](#_Toc1468915877)

[7 Discussion of Results 13](#_Toc1979672346)

[8 Related projects and existing code: 13](#_Toc1460384826)

[9 Your Feedback 13](#_Toc2124129783)

[10 References 14](#_Toc108754743)

# Introduction

A Minesweeper AI player can be designed to play Minesweeper efficiently without human interface, minimizing mistakes and maximizing success. High-success-rate AI players employ various techniques, including propositional logic, probability analysis, Bayesian network and heuristic algorithms to maximize the AI agent’s winning rate. The Minesweeper game has simple mechanics, although it presents a challenge due to it’s hidden information and randomized layout. This makes it an ideal challenge for testing and developing the AI to see if it is capable of reasoning under uncertainty.

# Problem Statement

This project is planning to utilize Knowledge Based Agents to play Minesweeper with propositional logic and probability method. The Knowledge body or agent then will be improved with Bayesian network and Constraint Propagation. The final goal is to improve the Agent’s efficiency and winning rate while minimizing unnecessary guesses playing Minesweeper. By integrating multiple AI strategies, the agent should be able to demonstrate an ability to shift between logic and probabilistic estimation based on the game state.

# Challenges

The primary objective of this project is to develop a AI player for Minesweeper game using proportional logic to enhance its ability in strategic decision making. The project plans to address some key challenges and build on the AI’s performance through adaptability, efficiency and logical reasoning.

1. Build an AI player using propositional logic
2. Improving the AI performance and efficiency
3. Improve how it makes decisions, and help it figure the game better.
4. Come up with better ways to handle conflicts so it does not get stuck.
5. Speed up its decision-making with better methods
6. Enable the AI’s adaptability to pick the best moves using heuristics and probabilities instead of guessing.
7. Help make the AI more adaptable and learn from advanced logic and strategies at solving the game.

## RELATED WORKS

3.1.1 Propositional Logic

The game board can be represented using propositional logic, where each tile's state which is mine or safe is expressed as a logical formula. We are given an initial bag of strings, called axioms, to inference the game with propositional logic. Constraint Satisfaction Problem (CSP) solvers use these formulas to deduce safe moves and mine locations. When a tile x is probed, x constrains the number of mines contained in its 8 neighbor cells. Suppose x contains a number n such that n ∈ {1, . . . , 8}. n is the number of mines in the x’s neighbors (Becerra, 2015). If a tile shows "1" and touches only one unknown tile, then that tile must contain a mine (Russell & Norvig, 2016). A value of 0 indicates the neighbor cells is mine-free. The integer n will be denoted as the label of a constraint. These rules allow the player to safely sweep the left tiles which do not have a mine.

3.1.2 Probability-Based Decision Making

When logic alone is not enough, probability models estimate the likelihood of mines in uncertain situations is introduced. Monte Carlo Trees simulations and Bayesian inference are the popular models used to refine the estimations in Mineswepeer (Sinha et al., 2021). Upper Confidence Tree (UCT) is combined with a Monte Carlo simulation to work on problems with no expert information or evaluation function. By using the heuristic strategies in the tree part of the algorithm or in the leaf evaluation; it makes improvement by simulating complete runs (Sebag, & Teytaud 2012).

3.1.3 Overlapping

When sentences in the knowledge base overlap, the agent infers additional mines by analyzing the relationships between overlapping sets.

3.1.4 Pattern Recognition

A pattern is a common arrangement of mines with fixed solutions. The pattern in Minesweeper is consistent with a grid pattern specified mines, they are the number of mines adjacent to a grid position, and the unknowns is an NP-complete problem (Kadlac, 2003). Learning patterns for the agent is time saving because solving the pattern during the game is slowing in response. The two most famous patterns are 1-2-1 and 1-2-2-1, by recognizing those common patterns in Minesweeper we can reduce the computation complexity and help our agent make faster decisions.

## IMPORTANCE AND IMPACTS

In this project, we are taking up the challenge of creating AI agent to play the stochastic puzzle minesweeper with winning odds matching human performance. The purpose of the project is to employ different AI techniques to build computational lite AI agent for playing minesweeper game.

We are implemented the AI agent using logical move using technique as propositional logic, constrain satisfaction problem, overlapping mines and overlapping safe cells and out of moves, AI Player uses probability methods as Bayesian inference up to mine probability of 0.8 and then monte Carlo with 10,000 simulations for last few final moves.

The main limitation or challenges we faced while we attempting to achieving human level winning odds in three different difficulty levels and program needs to lite and not requires lot of computation power or time. To overcome the challenge, we are decided to use basic AI technique to solve the simple, straightforward safe moves 80 % and then create hybrid solutions to solve remaining hard 20% of move using constrain satisfaction problem, overlap mine and probabilities methods.

# Data Collection and Preprocessing

The Minesweeper is imported from the Pygame. Beginner, Intermediate, and Difficult are the three different levels of difficulties used to run the test.

The data using for results are generated from the Python program and plotted for further analysis.

# Methodology

As minesweeper Puzzle is stochastic in nature, Logical technique is not sufficient to finish a game, so whenever there is no possible logical move we are using probability move. We are using Bayesian (requires less computation capacity) when unrevealed cells are greater than 10 and using Monteo Carlo (requires less computation capacity) for the final sets of moves

* Based on the analysis on the previous paper where AI agent is implemented for minesweeper, In most papers they took a logical move and probability move.
* All the previous paper implements propositional logic and used either Baysein or Monte Carlo probability methods.
* To increase the winning odds the reinforcement learning in deep Q or double Q is employed.
* As Minesweeper is stochastic, we have probability moves at end of the hybrid approach which will give a move for sure unlike logical approach.
* The game interface is designed using pygame.

**5.1 Game Setup**

The standard bord is set as 9\*9 with 10 mines as beginner difficult level. Minesweeper game code with AI-Play, AIPlay-100 Games and Reset buttons as show in Figure 1 and its corresponding Minesweeper AI move are coded in runner.py.

The Minesweeper AI move implemented for the minesweeper game using logical move using technique as propositional logic, constrain satisfaction problem, overlapping mines and overlapping safe cells and out of moves, AI Player uses probability methods as Bayesian inference up to mine probability of 0.8 and then monte Carlo with 10000 simulations for last few final moves

A screenshot of a game

AI-generated content may be incorrect.

Figure 1. Minesweeper Game UI Set up

5.2 High level algorithms

Below is high level algorithm for AI player Implementation

* + Initialization (Height, Width, Mines):
  + Initialize class with height and width of the board.

moves\_made:

* + Initialize a set to store all completed moves.
  + mines: Initialize a set to store identified mines.

mark\_mine(cell):

* + Adds identified mine cell( i, j) to mines set.

mark\_safe(cell):

* + Adds identified safe cell( i, j) to safe\_moves set.

add\_knowledge (cell, count):

* + Add the cell to moves\_made set.
  + Mark cell as safe.
  + Identify all adjacent unknown cells (not already revealed).
  + Create a knowledge sentence: (adjacent\_unknown\_cells, count) and add to knowledge.
  + Call update\_knowledge() to refine the knowledge base.

update\_knowledge():

* + Iteratively process each sentence (cells, count) in knowledge:
  + Identify already known mines in the sentence.
  + Identify already known safe cells in the sentence.
  + If the number of known mines equals count, mark all other cells as safe.
  + If the number of unknown cells equals count, mark all cells as mines.
  + Remove fully resolved sentences from knowledge.
  + Repeat the process until no new information is inferred.

5.3 Propositional Logic

Two key rules:

1. If the number of unknown cells = the number of mines, then all those cells are mines.

  A ∨ B ∨ C, and count = 3 ⇒ A ∧ B ∧ C are all mines.

2. If count = 0, then all cells are safe.

  ¬A ∧ ¬B ∧ ¬C (none are mines).

Table 1. Propositional Logic Condition, Logical Form and Interpretation

|  |  |  |
| --- | --- | --- |
| **Condition** | **Logical Form** | **Interpretation** |
| count == 0 | ∀x ∈ cells, ¬*M*(x) | All are safe |
| len(cells) == count | ∀x ∈ cells, *M*(x) | All are mines |

This propositional logic is deterministic and allows for definite safe/mine deductions without probabilities.

5.4 Constrain Satisfaction Problem

Constraint Satisfaction Problem (CSP) logic to deduce new safe cells or mines by reasoning over the knowledge base as shown in Table 2.

Table 2. CSP Condition, Logical Form and Interpretation

|  |  |  |
| --- | --- | --- |
| **Condition** | **Logical Form** | **Interpretation** |
| S1 ⊂ S2 | S2 – S1 = c2 - c1 | Deduce subset sentence from the main sentence |
| count == 0 | ∀x ∈ cells, ¬*M*(x) | All are safe |
| len(cells) == count | ∀x ∈ cells, *M*(x) | All are mines |

5.5 Heuristic Algorithm

The partial\_overlap\_inference() function in your MinesweeperAI is designed to apply propositional logic on partially overlapping sentences as shown in Table 3. If both sentences are true, their overlap contributes to both counts, so we subtract it twice to avoid double-counting.

Example:

 {A, B, C} = 2 → Two of these are mines

 {B, C, D} = 2 → Two of these are mines

 Their overlap is {B, C}   
 Their unique parts are {A} and {D}

 New sentence: {A, D} = 0, So A and D must be safe.

Table 3. Heuristic Algorithm Condition, Logical Form and Interpretation

|  |  |  |
| --- | --- | --- |
| **Condition** | **Logical Form** | **Interpretation** |
| S1 & S2 overlap | New\_cells = (S1 \ S2) ∪ ( S2 ∖ S1)  New\_count = c1 + c2− 2 x (overlap)  New\_sentence = new\_cells⇒new\_count | Deduce new sentence from overlap |
| count == 0 | ∀x ∈ cells, ¬*M*(x) | All are safe |
| len(cells) == count | ∀x ∈ cells, *M*(x) | All are mines |

5.6 Bayesian Reasoning

In the bayesian\_inference method of the MinesweeperAI class, the code uses a simplified Bayesian-style heuristic.

Formula in Code:

    cell\_probs[cell] = (cell\_probs[cell] + likelihood)/2

This is a weighted average of:

A prior probability of the cell being a mine:

  prior = remaining mines / unrevealed cells

A likelihood estimate based on knowledge sentences:

  likelihood for cell = count / number of unknown cells in sentence

Each cell’s probability is updated by averaging this likelihood with the prior.

The Final Safe Move is decided by the cell with minimal probability to be a mine.

  safe\_move = min (cell\_probs[unreveal(cell)])

5.7 Monte Carlo Simulation

The Monte Carlo simulation to estimate the safest cell to click based on the knowledge when no deterministic (logic-based) move is possible. The steps of the Monte Carlo simulation is shown as follows.

Table 3. Monte Carlo Simulation Steps

|  |  |
| --- | --- |
| **Steps** | **Purpose** |
| Simulate consistent mine placements 10,000 times with the knowledge | Model uncertainty when logic is insufficient |
| Count how often each cell is safe | Estimate safety probability |
| Pick highest-scoring cell | Make the best probabilistic move |

Choose the Cell with the Highest Score as the safe cell to make the move. Mote Carlo is only used at the final steps of the game as it requires more computational power to run 10,000 simulations.

5.8 Smart Move

Smart Move is the uplevel algorithm to have the AI agent switch between all different AI techniques to define the safe cells and mines to increase the winning rates as shown in Figure 2. This is hybrid move, which employes Propositional logic as first approach. When propositional logic didn’t return any move then it will check the CSP approach and then identify overlap mine. When all three approach is not returned any possible move. The smart move checks if there are still games to play, if yes then it will resort to probability moves.

In probability moves, smart moves employ Bayesian inference to identify a move at an early stage of the game. For later stage it uses monte Carlo probability with 10,000 simulations to identify the safe move.

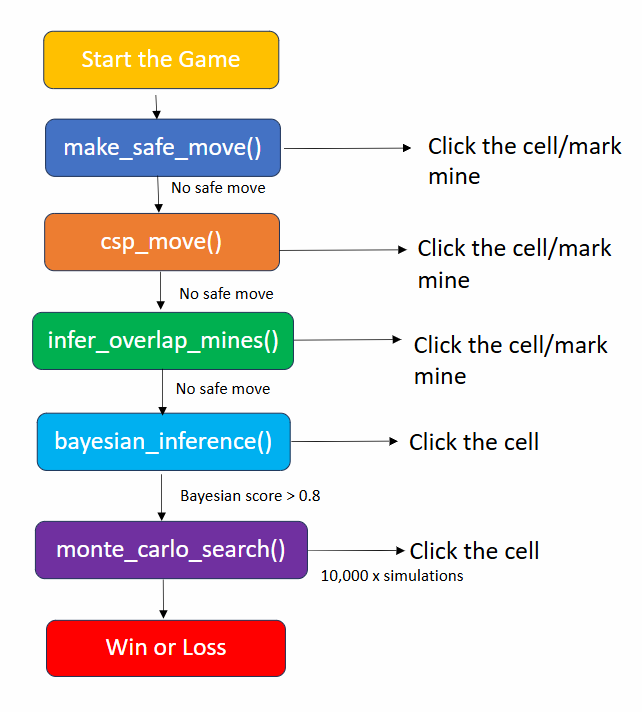


Figure 2. Smart Move Architecture

# Results and Interpretation

We have the Minesweeper settings according to the standard Beginner, Intermediate, Difficult as shown in Table 4. After that the AIPlayer and AIPlayer-100 are activated to generate the winning rate for 100 games. As the data show below we can see the winning rate for the Beginner is 91%, for the Intermediate level is 71%, for the Difficult level is 22%.

Table 4. Minesweeper AI Winning Rate at Different Difficult Levels

|  |  |  |  |
| --- | --- | --- | --- |
| Minesweeper AI Performance | Beginner | Intermediate | Difficult |
| Window Size | 9 x 9 | 16 x 16 | 30 x 16 |
| Mines | 10 | 40 | 99 |
| Win Out of 100 Games | 91 | 71 | 22 |

As the difficulty level increases from Beginner to Difficult, the winning rate dropped. The screen shoots of the Minesweeper game winning result are provided as in Figure 3.

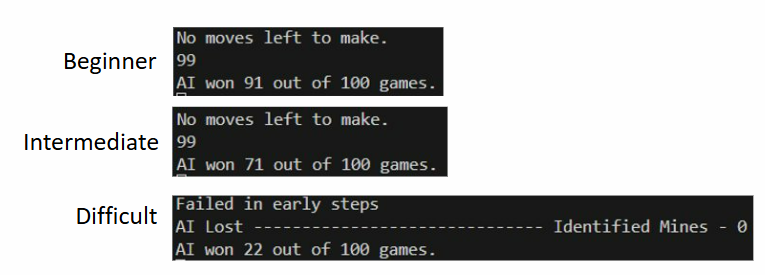


Figure 3. Screen shoots of the Python Minesweeper winning print out

We also wanted to know the impact of the different techniques including propositional logic, constrain satisfaction problem, heuristic algorithms, Bayesian Network, and Monte Carlo Simulation on game winning rate. The basic group is the Safe Move (propositional logic) with Bayesian, then we adde the CSP, after that we introduced overlap, and finally we added the Monte Carlo. With this consequence we can investigate each technique’s impact on game winning rate.

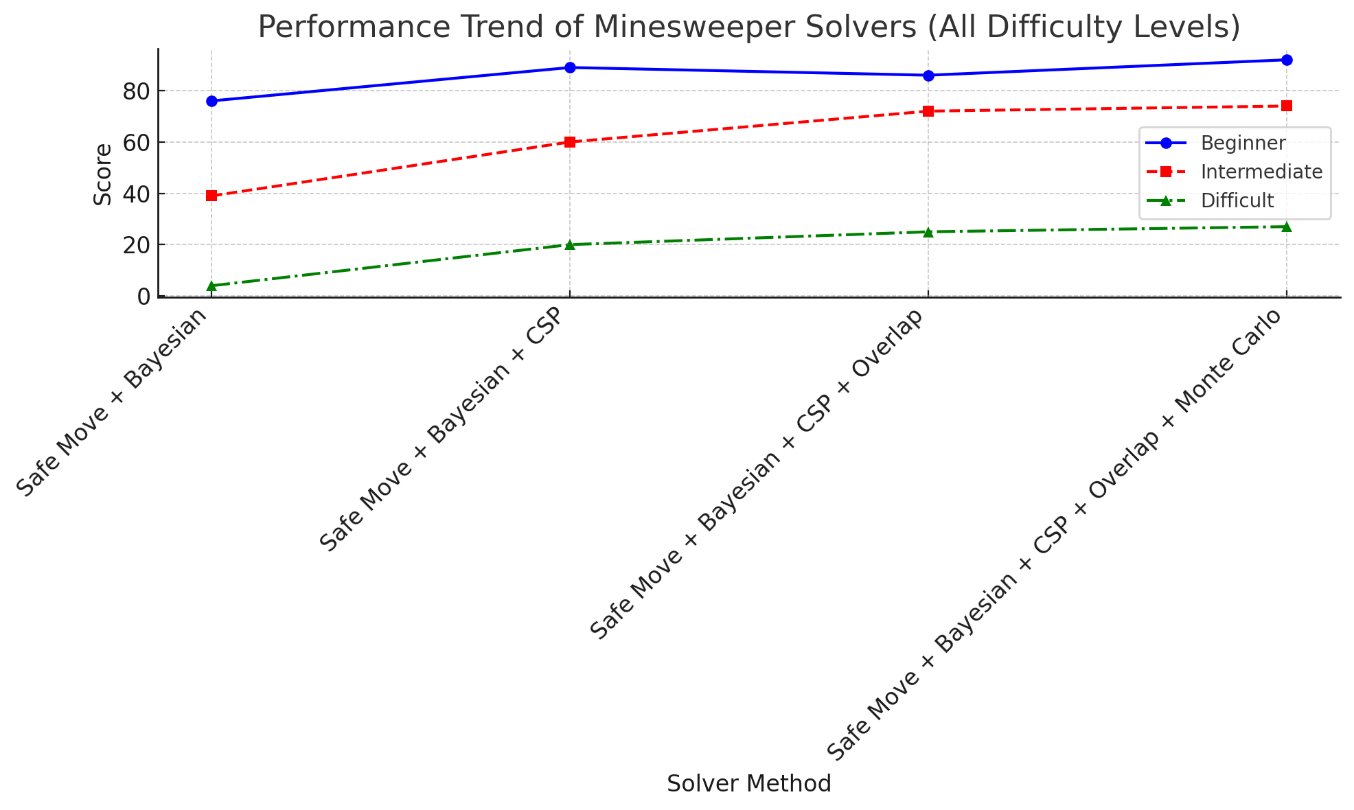


Figure 4. Performance Trend of Minesweeper AI Solvers

The different Minessweeper AI solvers performance data are shown in the Figure 4 and Table 5. At the Beginner level, Safe Move with Bayesian winning rate is 76%, after we add the CSP is 89%, after we introduced overlap it is 86%, and finally we added the Monte Carlo and winning rate is 92%. At the Intermediate level, Safe Move with Bayesian winning rate is 39%, after we add the CSP is 60%, after we introduced overlap it is 72%, and finally we added the Monte Carlo and winning rate is 74%. At the Difficult level, Safe Move with Bayesian winning rate is only 4%, after we add the CSP is 20%, after we introduced overlap it is 25%, and finally we added the Monte Carlo and winning rate is 27%.

Table 5. Performance Data of Minesweeper AI Solvers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Performance Trend of Minesweeper Solver | Safe Move + Bayesian | Safe Move + Bayesian + CSP | Safe Move + Bayesian + CSP + Over\_lap | Safe Move + Bayesian + CSP + Over\_lap + Monte Carlo |
| Beginner | 76 | 89 | 86 | 92 |
| Intermediate | 39 | 60 | 72 | 74 |
| Difficult | 4 | 20 | 25 | 27 |

# Discussion of Results

The winning rates decrease as the game difficulty increases from beginner (91%), to intermediate (71%), to difficult (22%). Hybrid techniques have improved the winning rate of the game. If we only implement the basic techniques which are Prepositional Logic and Bayesian, the winning rate is low. After we add CSP, Overlapping, and Monte Carlo the winning rate increases at different difficulty levels. At beginner level the winning rate increases from 76% to 92%, at intermediate level the winning rate increases from 39% to 74%, At difficult level the winning rate increases from 4% to 27%

We introduced a Minesweeper AI player with mixed logic techniques and probability inference. The logic techniques include propositional logic, constrain satisfaction problem, and heuristic algorithms to label the safe cells and identify mines. When there is no logic move left, we employee the probability analysis including Bayesian Network, and Monte Carlo Simulation to decide the safe move. The Minesweeper AI player can reach a winning rate of 92% at the highest.

To improve AI player performance, we plan to implement either pattern recognition or reinforcement learning. However, the reinforcement learning needs a training process and will slow down the gaming solving process and increase the computation complexity.

# Related projects and existing code:

[https://cs50.harvard.edu/summer/ai/2022/projects/1/minesweeper](https://cs50.harvard.edu/summer/ai/2022/projects/1/minesweeper )

# Your Feedback

Dr. Guang Yang is a very responsible professor who have organized the AI 801 Foundation of AI very well. With 14 weeks of studies, we had learned how to apply AI to solve real problems in the industry. This final project is an extend of our knowledge and applying the skills of Minesweeper which is challenging. Through the process we improved the winning rate graduate by introducing logic and rules into the AI.

# References

1. https://cs50.harvard.edu/summer/ai/2022/projects/1/minesweeper/

2. Becerra, D. J. (2015). Algorithmic approaches to playing Minesweeper (Doctoral dissertation).

3. Kadlac, M. (2003). Explorations of the minesweeper consistency problem. Oregon State University, Research Experiences for Undergraduates.

4. Kaye, R. (2000). "Minesweeper is NP-Complete" – Theoretical complexity of Minesweeper.

5. Mehta, A. (2021). Reinforcement learning for constraint satisfaction game agents (15-puzzle, minesweeper, 2048, and sudoku). arXiv preprint arXiv:2102.06019.

6. Russell, S. J., & Norvig, P. (2016). Artificial intelligence: a modern approach. Pearson. Chapters on logic, CSPs, and probabilistic reasoning.

7. Sebag, M., & Teytaud, O. (2012). Combining myopic optimization and tree search: Application to minesweeper. In LION6, Learning and Intelligent Optimization (Vol. 7219, pp. 222-236). Sringer Verlag.

8. Sinha, Y. P., Malviya, P., & Nayak, R. K. (2021). Fast constraint satisfaction problem and learning-based algorithm for solving Minesweeper. arXiv preprint arXiv:2105.04120.

9. Sutton & Barto, "Reinforcement Learning: An Introduction" (2018) discusses Monte Carlo methods for estimating probabilities.

10. https://www.geeksforgeeks.org/inference-in-ai/