Knowledge-Based Systems 246 (2022) 108630

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A solver of single-agent stochastic puzzle: A case study with Minesweeper☆

Chang Liu [a,*], Shunqi Huang [a], Gao Naying [a], Mohd Nor Akmal Khalid [a,b], Hiroyuki Iida [a]

[a] Japan Advanced Institute of Science and Technology, School of Information Science, 1-1 Asahidai, Nomi, 923-1211, Ishikawa, Japan
[b] Universiti Sains, School of Computer Sciences, Georgetown, 11800, Pulau Pinang, Malaysia

ARTICLE INFO

Article history:
Received 20 November 2021
Received in revised form 2 March 2022
Accepted 19 March 2022
Available online 28 March 2022

Keywords:
Single-agent game
Stochastic puzzle
Minesweeper
Constraint satisfaction problems
Gauss–Jordan elimination

ABSTRACT

People have enjoyed solving puzzles for decades because of the challenge and the satisfaction derived from solving problems. However, although previous researches focused on the puzzle's complexity, strategy, and solving automatically, few studies worked on sorting out puzzles from a solvability way related to the stochastic elements among the solving process. The contribution of the study is twofold. Firstly, a single-agent stochastic puzzle definition is established via Minesweeper testbed, a well-known puzzle synonymous with Microsoft Windows. Secondly, this study proposes an artificial intelligence (AI) solver based on the obtained information on the board, called the 'PAFG' strategy, which stands for the primary reasoning, the advanced reasoning, the first action strategy, and the guessing strategy. The first two strategies take advantage of knowledge-based rules and linear system transformation (Gauss–Jordan elimination algorithms) to determine the probability of making a move independently. The last two strategies explore the beginning and ways to determine hidden puzzle states to enhance the winning rate of the AI solver. The experimental simulation of various configurations and the AI solver with PAFG strategy yielded a high-level winning rate of 96.4%, 86.3%, and 45.6% for the $9 \times 9|10$, $16 \times 16|40$, and $16 \times 30|99$ Minesweeper board configuration, which is comparable to the state of the art study. Thus, such an AI solver could contribute to classifying single-agent stochastic puzzles and establishing the boundary of the puzzle-solving and game-playing paradigm.

1. Introduction

Proposing an efficient puzzle solver has been a significant research paradigm in Artificial Intelligence (AI). Since the work by Allis et al. [1] in 1994, efficient solving mechanisms and algorithms had been exclusively conducted on the domain of two-player zero-sum games. Based on such domain, various methodologies have been proposed which take advantage of the deterministic nature of such games, which is associated with predicting the game-theoretic value [2,3] and determining optimal strategy [4].

Due to the wide variety of puzzles of different properties, many research efforts had been dedicated to investigating the computational complexity [5] and the possible classification of puzzles [6]. Moreover, a recent attempt at classifying puzzles had been mathematically formulated by simplifying the essential components of the puzzle into a Boolean satisfiability problem (SAT) or Constraint Satisfaction Problem (CSP) to determine classes of the puzzle [7]. Thus, a problem-specific Monte-Carlo tree search (MCTS) can effectively be applied to the respective puzzle classes based on such mathematical definitions. However, such an investigation was still limited to deterministic puzzles.

As defined by Kiarostami et al. [7], there are three types of puzzle classes: class A, class B, class AB, and class C. The class A puzzle is a puzzle that can be statistically solved with simple constraints. Meanwhile, class B puzzle is a puzzle that is directly affected by the step or sequence of moves with an additional time-related dimension. Moreover, the class AB is the combination of class A and class B where the solver can address both classes. Finally, class C is the puzzle that is outside of the other classes and has at least one random feature and/or inputs.

The 2048 and Minesweeper are examples of the class C puzzle, since the former had random input variables in every state while the latter contains random placement of hidden mines. Recently, a 2×2 version of 2048 had been solved by Primanita et al. [8]. A similar puzzle, namely Minesweeper, corresponds with a single-agent incomplete-information game [9] that fits the criteria of the

☆ This research is funded by a grant from the Japan Society for the Promotion of Science, in the framework of the Grant-in-Aid for Challenging Exploratory Research (Grant Number: 19K22893).

* Corresponding author.
E-mail addresses: liuchang@jaist.ac.jp (C. Liu), s1910413@jaist.ac.jp (S. Huang), s2120016@jaist.ac.jp (G. Naying), akmal@jaist.ac.jp (M.N.A. Khalid), iida@jaist.ac.jp (H. Iida).

https://doi.org/10.1016/j.knosys.2022.108630

class C puzzle, which had been recently investigated to determine the reason for its attractiveness [10]. With the growing community around solving Minesweeper, various AI agents or solvers have been introduced in the literature [11–16].

Nevertheless, a clear definition of puzzle categories is needed from the perspective of its solvability. For such purpose, the Minesweeper puzzle was utilized as the benchmarking testbed. In such a condition, what distinguishes a deterministic puzzle from other types of puzzle, i.e., a stochastic puzzle? Therefore, this study proposes a definition of a stochastic puzzle from its solvability, which forms the foundation for the proposed AI solver. Moreover, the proposed AI solver takes advantage of both the deterministic and stochastic elements of the puzzle, which called PAFG: the primary reasoning strategy ("P"), the advanced reasoning strategy ("A"), the first action strategy ("F"), and the guessing strategy ("G"). Such strategies also combine mathematical model, knowledge-driven rules, and linear transformation to provide conducive moves in solving Minesweeper, comparable to previously proposed Minesweeper AI solvers.

2. Related works

2.1. A stochastic puzzle testbed: Minesweeper

Minesweeper is a classic single-agent incomplete-information game, which gains its popularity as it became a feature of Microsoft Windows in 1989 [9,17]. In the 1980s, various Minesweeper-type video games were designed, such as Mined-Out, Yomp, and Relentless Logic.[1] After that, Minesweeper was integrated into the entertainment pack in many generations of Microsoft Windows, like 3D, hexagonal, triangular, and multiple Minesweepers.

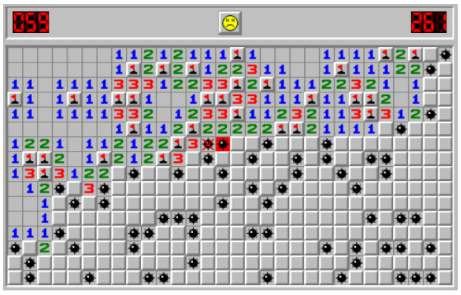
The goal of Minesweeper is to open all non-mine hidden cells on the board and flag all the mines on the board. If the player opens instead of flagging a mine cell incorrectly, the game is lost. For example, Fig. 1(a) and (b) show a losing state and a winning state of Minesweeper ($16 \times 30|99$ mines), respectively. In this paper, we use three classic Microsoft Minesweepers as the testbed: beginner ($9 \times 9|10$ mines), intermediate ($16 \times 16|40$ mines), and expert ($16 \times 30|99$ mines). To simplify its reference throughout the process, these three puzzles were designated as board labeled as X, Y, Z, respectively.

The discrete model-based cellular automaton was first used to solve Minesweeper proposed by Adamatzky [18] in 1951. Various distribution and mines density were recorded, and several agents would automatically solve the problem mapped to specific networks by making random choices using an adaptive algorithm. Besides, Kaye [19] found that solving Minesweeper is an NP-complete problem which means that Minesweeper is hard to be solved in polynomial time. Meanwhile, Studholme [11] formulated Minesweepers as CSPs with finite constraints or limitations to be satisfied. A heuristic strategy with a backtracking algorithm was used to find the best guess in solving the game.

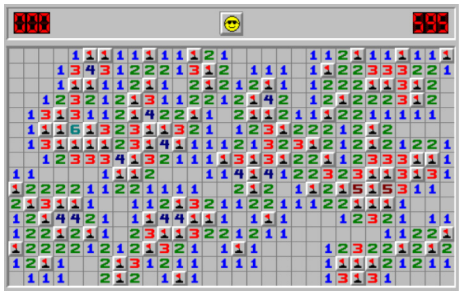
2.2. Minesweeper's complexity analysis

From the computational complexity perspective, previous studies found that Minesweeper is too hard to solve in polynomial time. Moreover, as one of the NP-complete problems, Minesweeper is the most difficult NP problem as found by Kaye [19]. From Minesweeper consistency problem, given a game of Minesweeper configuration and a board state with some visible numbers and flagged mines, does there exist a possible distribution of unknown mines to satisfy known information? Arora and Barak [20] proposed that the Boolean satisfiability problem (SAT) could reduce to Minesweeper consistency problem, which was described as Boolean formula (i.e., all variables take TRUE or FALSE assignments). Furthermore, based on a theorem by Cook [21] that Boolean satisfiability problem (SAT) is NP-complete, which proved that the Minesweeper consistency problem is NP-complete. Then, solving the Minesweeper consistency problem provides a way to Minesweeper inference problem. For instance, given a game of Minesweeper configuration and a board state, does a hidden cell exist that the player can infer undoubtedly? Arora and Barak [20] and Scott et al. [22] studied Minesweeper inference problem from the perspective of computational complexity theory. They proved that the Minesweeper inference problem is a co-NP-complete problem (the complements of NP problem), which is the most challenging problem in co-NP and reduces the complement of SAT to the Minesweeper inference problem.

Three possible correlations between P, NP, NPC, and co-NP are shown in Fig. 2. Polynomial-time (P) issues are those for which a method exists to solve them in polynomial time. Nondeterministic polynomial time (NP) problems exist when an algorithm validates the response "YES" in polynomial time for problem instances. A problem p is an NP-complete problem if and only if p is an NP problem and every NP problem can be reducible to p in polynomial time [20], which means that NPC problems are the most challenging problems in NP problems. Moreover, co-NP problems are complements of NP problems. In other words, co-NP problems are those for which a polynomial-time solution exists to validate the answer "NO" for problem instances. A problem p is a co-NP-complete problem if and only if it is a co-NP problem, and every co-NP problem can be reduced to p in polynomial time, implying that co-NP-complete problems are the most challenging co-NP problems.



(a) A losing state of Minesweeper



(b) A winning state of Minesweeper

Fig. 1. Two final states of Minesweeper $16 \times 30|99$ mines, the purpose is to find all hidden mines on the board without opening them. (a) is a losing state because the player opened on a mine in the game process, (b) is a winning state while the player revealed all mines on the board.

[1] http://www.minesweeper.info/wiki/Windows_Minesweeper#Windows_3.1.

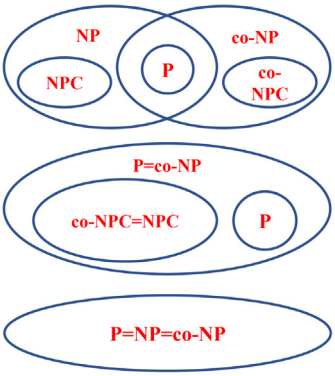


Fig. 2. An illustration of three possible relationships among P, NP, NPC, and co-NP.

2.3. The solvability of single-agent game: Evidence from Minesweeper

Generally, defining the solvability of a game depends on the observable outcome of the game. In the context of two-player zero-sum games, Allis et al. [1] has defined solvability from the perspective of game-theoretic values, which have three different levels: ultra-weakly solved, weakly solved, and strongly solved. Such a solvability definition revolves around the legal positions (or moves) made in the game (initial or all). As a result, a varying degree of game-theoretic value is achieved under reasonable resources. Such a degree of game-theoretic determination and consideration of the legal positions were the features that differentiate the three-level of solvability of the two-player zero-sum games. Based on such understanding, this study develops the definition of a single-agent game's solvability based on such a foundation.

In the context of the solvability definition of two-player zero-sum games, solving the game corresponds to finding a solution via a solution tree using tree search methods (such as AND/OR tree search or equivalent). In this study, the concept of solving games is extended, which focuses on stochastic single-agent games. To define stochastic single-agent games, the deterministic single-agent games must first be established from the solvability perspective. As such, the solvability of a deterministic single-agent game can be defined as finding a solution with 100% probability for any initial position. Furthermore, as discussed by Kiarostami et al. [7], a deterministic single-agent game is solvable when a solution can be statistically determined with simple (class A) or time/state-dependent (class B) constraints.

A survey conducted on the different single-agent games had described its solvability with features such as random, and hidden information is called a stochastic game [5,7]. For example, Minesweeper and Tetris are stochastic single-agent games, while N-puzzle is a deterministic single-agent game. As puzzle games evolved, stochastic single-agent games are NP-hard problems such as Threes, and 2048 puzzle [23]. For a stochastic puzzle, the solvability of such a puzzle is defined as finding a solution with less than 100% probability but also being more than 0%.

Let p be the probability of a win (or winning rate) of a single-agent game. Thus, a stochastic single-agent game is considered a 'puzzle' if and only if $0 < p < 1$ in finding a solution for any initial position. Otherwise, it is regarded as a 'game'. These two distinction is established as the possible border between puzzle-solving and game-playing. In this study, puzzle-solving is regarded as exhibiting more than 0% possibility of finding a solution for any initial position of a single-agent game. Meanwhile, game-playing is regarded as exhibiting no certainty of a solution to be found, as shown in (1).

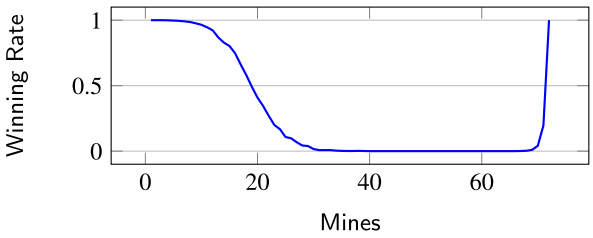


Fig. 3. The winning rate (p) of the 9×9 board size Minesweeper is based on the number of mines $M \in [1, 72]$, with 2000 runs per mine. The winning rate $p = 1$ for $M \leq 3$ indicates a deterministic puzzle; $0 < p < 1$ for $M \in [4, 39]$ and $M \in [67, 71]$ indicates a stochastic puzzle; otherwise, $p = 0$ for $M \in [40, 66]$ indicates a game.

$$p = \begin{cases} 1, & \text{deterministic puzzle} \\ 0 < p < 1, & \text{stochastic puzzle} \\ 0, & \text{game} \end{cases} \tag{1}$$

A single-agent AI solver of Minesweeper was conducted in 9×9 board for 144,000 runs, where the mines are $1 \leq M \leq 72$. Fig. 3 shows the winning rate $p = 1$ when $M \leq 3$; thus, indicating that such a situation corresponds to a deterministic puzzle. Relative to the definition given by Allis et al. [1], such a situation indicates a strongly solved instance since the strategy adopted by the AI agent can determine the game-theoretic value for all positions. As the number of mines increases, the winning rate decreases for $M \in [4, 39]$, which corresponds to a weakly solved situation. In other words, the AI agent can determine at least some game-theoretic value for an initial position since some uncertainty occurred that makes certain positions in the game unsolvable. When $M \in [40, 66]$, $p = 0$ that a solution is unavailable, a puzzle becomes a game. Finally, $p \to 1$ for $M \in [67, 72]$ as available information is sufficient to determine a solution (or determine game-theoretic game) after opening only several cells (similar situation as a strongly solved instance).

The experiment conducted on the 9×9 Minesweeper, which 37.5% (54,000 instances) of the total instances are unsolvable. Note that such instances may vary depending on different factors; nevertheless, they constitute what can be defined as the "uncertainty" element of the single-agent game.

3. Methodology and algorithm

The overall of the PAFG strategy is given by Algorithm 1. The first cell is opened according to the first action rule, which ensures the first opened cell and the surrounding cells are safe. Then, the primary reasoning strategy is simple and effective, the advanced reasoning strategy is used based on the primary reasoning strategy, and the problem switches to finding invariables in a linear system of equations. Moreover, the Gauss–Jordan elimination method is used to get general solutions, and some invariables can be found since the solutions satisfy certain constraints. Furthermore, the guessing strategy is based on the result of Gauss–Jordan elimination, and the binary tree search based on the linear system model is used, where the binary tree is pruned according to the constraints on the solutions of linear systems.

The time complexity for the primary reasoning strategy is $O(n)$, n is the number of unsolved number cells. For the advanced reasoning strategy, the time complexity is $O(n^3)$, and n is the number of variables in the system of equations. Moreover, the time complexity for the guessing strategy is $O(2^n)$, where n is

Algorithm 1 The Overall Algorithm

Input B: game board
open the first cell according to the first action rule
while the game is not end **do**
  $S_1, S_2 \leftarrow PRS(B)$
  if $S_1 and S_2$ are not empty **then**
    open the cells in S_1
    flag the cells in S_2
  end if
  $S_1, S_2, C, V \leftarrow ARS(B)$
  if $S_1 and S_2$ are not empty **then**
    open the cells in S_1
    flag the cells in S_2
  end if
  $S_1, S_2 \leftarrow GS(B, C, V)$
  open the cells in S_1
  flag the cells in S_2
end while

the number of variables, and setting the number of free variables $n \leq 20$. The size of the entire search tree is capped at 2^{20}, and tree pruning dramatically reduces the size of the search tree; thus, significantly reducing the expected complexity.

3.1. Building Minesweeper AI

The configuration of Minesweeper is denoted by $R \times L|b$, where $R \times L > b$ and $R, L, b \in \mathbb{N}^*$, R and L are the number of rows and columns on the board, respectively, and b is the number of mines.

Definition 1 (*Cell Position*)**.** For any cell c, it can be represented by a pair of its unique position (x, y), i.e., $c = (x, y)$, where $1 \leq x \leq R$, $1 \leq y \leq L$ and $x, y \in \mathbb{N}^*$. x is the row number, and y is the column number.

According to Definition 1, the cell on the top left corner of Fig. 4(a) can be represented as $c = (1, 1)$. We can indicate that for any two different cells (x_1, y_1) and (x_2, y_2), $(x_1, y_1) \neq (x_2, y_2)$ is equivalent to $x_1 \neq x_2$ or $y_1 \neq y_2$.

Definition 2 (*Cell Type*)**.** For any cell c, the cell type $I(c)$ is one of a number from -1 to 10, i.e., $I(c) \in \{-1, 0, \ldots, 10\}$. The meaning is as follows:

- $I(c) = -1$: c is an exploded mine.
- $I(c) = 0$: c is the number "0" displayed as a blank opened cell.
- $1 \leq I(c) \leq 8$: c is a number that equals $I(c)$.
- $I(c) = 9$: c is a hidden cell.
- $I(c) = 10$: c is a flagged cell.

According to Definition 2, $I(\cdot)$ is a function that the input is the position of a cell, the output is a type integer, and the type of a cell is unique.

Definition 3 (*Cell Set*)**.** The Cell set C is the set of all cells on the board, i.e., $C = \{(x, y) \mid 1 \leq x \leq R,\ 1 \leq y \leq L,\ x, y \in \mathbb{N}^*\}$.

Definition 4 (*Neighborhood*)**.** For a cell $c = (x_0, y_0)$, the neighborhood $N(c)$ is the set whose elements are cells that are adjacent or diagonally adjacent to c, i.e., $N(c) = \{(x, y) \mid x_0 - 1 \leq x \leq x_0 + 1,\ y_0 - 1 \leq y \leq y_0 + 1,\ (x, y) \in C,\ (x, y) \neq (x_0, y_0)\}$.

According to Definition 4, we can see $N(\cdot)$ as a function that the input is the position of a cell, the output is a set of positions of surrounding cells, noted that two different cells cannot have the same neighborhood.

Definition 5 (*Corner Cell*)**.** A cell c is a corner cell if $|N(c)| = 3$, where $|N(c)|$ is the cardinality of $N(c)$.

Definition 6 (*Border Cell*)**.** A cell c is a border cell if $|N(c)| = 5$, where $|N(c)|$ is the cardinality of $N(c)$.

Definition 7 (*Unsolved Number Cell*)**.** A cell c is an unsolved number cell if $1 \leq I(c) \leq 8$, and there exists a cell $c^* \in N(c)$ such that $I(c^*) = 9$.

Definition 8 (*Unsolved Block*)**.** An unsolved block U is a set of cells that satisfies two conditions:

- For any $c \in U$, c is an unsolved number cell.
- Suppose that $|U| \geq 2$, for any $c_1 \in U$, there exists $c_2 \in U$ $(c_1 \neq c_2)$ such that $d_m(c_1, c_2) = 1$, where $d_m(c_1, c_2)$ is Manhattan Distance between c_1 and c_2, i.e., suppose that $c_1 = (x_1, y_1)$, $c_2 = (x_2, y_2)$, $d_m(c_1, c_2) = |x_1 - x_2| + |y_1 - y_2|$.

Definition 9 (*Frontier*)**.** For an unsolved block U, the frontier $F(U)$ is the set of cells that for any $c \in F(U)$, there exists an unsolved number cell $c^* \in U$ such that $c \in N(c^*)$.

Definition 10 (*Uninformed Set*)**.** A cell c is a uninformed cell if $I(c) = 9$ and for any $c^* \in N(c)$, c^* is not a number cell. The uninformed set is the set of uninformed cells.

Fig. 4(a) is an example of Minesweeper 9×9 board size. The gray blank cells have $I(c) = 0$. The cells with black "f" are flagged cells, and the cells with numbers are numbered cells. The set of orange blank cells is the frontier $F(U_1)$, the set of blue and purple blank cells is the frontier $F(U_2)$, the set of green and purple blank cells is the frontier $F(U_3)$, and the white cells are uninformed cells.

3.1.1. Primary reasoning strategy

The primary reasoning strategy has two rules. Firstly, the surrounding hidden cells are safe for any cell when their value equals the number of surrounding flagged mines. Secondly, the surrounding hidden cells are mines for any cell when the number of surrounding hidden cells equals its value minus the number of surrounding marked mines. These reasoning strategies were devised under the mathematical solving logic with the basic knowledge and rules, the general flow is described in Algorithm 2. For an unsolved number cell c, suppose that f^* and h^* are the set of flagged cells and hidden cells in $N(c)$, respectively, i.e., $f^* = \{c^* | I(c^*) = 10, c^* \in N(c)\}$ and $h^* = \{c^* | I(c^*) = 9, c^* \in N(c)\}$. Lines of $6 - 9$ are significant points in Algorithm 2, showing that if $|h^*| = I(c) - |f^*|$, for any $\bar{c} \in h^*$, $\bar{c}$ must be a mine.

Algorithm 2 Primary Reasoning Strategy

Input B: game board
Output S_1: the set of safe cells, S_2: the set of mine cells
1: **Function** PRS(B)
2: $S_1, S_2 \leftarrow \{\}$ ▷ empty set
3: **for** each unsolved number cell c on the board B **do**
4: $f^* \leftarrow$ the set of flagged cells in $N(c)$
5: $h^* \leftarrow$ the set of hidden cells in $N(c)$
6: **if** $|h^*| = I(c) - |f^*|$ **then**
7: $S_2 \leftarrow h^*$
8: **else if** $I(c) = |f^*|$ **then**
9: $S_1 \leftarrow h^*$
10: **end if**
11: **end for**
12: **Return** S_1, S_2
13: **end Function**

Since c is an unsolved number cell, $I(c)$ indicates the number of mines in $h^* \cup f^*$, the cells in f^* are flagged cells (or flagged

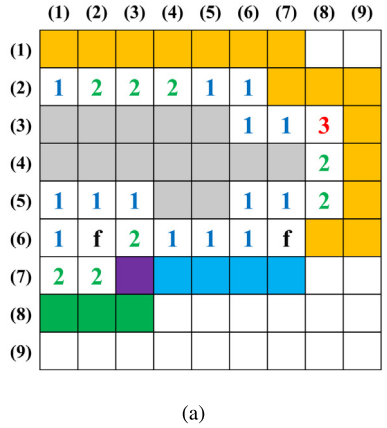


Fig. 4. (a) The basic information of the cells on the board for minesweeper ($9 \times 9|10$), there are three unsolved blocks on the board: $U_1 = \{(2,1),\ldots,(2,6),(3,6),\ldots,(3,8),(4,8),(5,7),(5,8)\}$, $U_2 = \{(6,3),\ldots,(6,6)\}$, $U_3 = \{(7,1),(7,2)\}$; (b) An illustration of primary reasoning strategy: The cells with "f" and numbers are flagged cell number cells, respectively.

mines), thus, there are $I(c) - |f^*|$ mines in h^*. There are $\binom{a_1}{a_2}$ way(s) to distribute mines, where $a_1 = |h^*|$, $a_2 = I(c) - |f^*|$, then, $\binom{a_1}{a_2} = 1$ since $|h^*| = I(c) - |f^*|$ (i.e., for any $\bar{c} \in h^*$, $\bar{c}$ must be a mine cell). If $I(c) = |f^*|$, then for any $\bar{c} \in h^*$, $\bar{c}$ must be a number. Since there are $I(c) - |f^*|$ mines in h^*, $I(c) = |f^*|$, there is no mines in h^*, i.e., for any $\bar{c} \in h^*$, thus, $\bar{c}$ must be a number cell. Fig. 4(b) is an example of primary reasoning strategy in Minesweeper case, where the subfigure (a) can be obtained since the cell $(2,1)$ with a number "3", so $(3,1)$ and $(3,2)$ are flagged mines, and (b) is obtained since the cell $(2,2)$ with a number "3", thus, $(1,3)$, $(2,3)$ and $(3,3)$ are safe.

3.1.2. Advanced reasoning strategy

The advanced reasoning strategy is based on the primary reasoning strategy with a deeper understanding of the mechanics of Minesweeper and knowledge-driven methods that exploit existing information. Firstly, the frontier division is conducted to get independent frontiers. Then, we see each hidden cell of this frontier as a Boolean variable for each independent frontier [11]. Subsequently, linear equations are proposed to find solution invariability in the constrained linear system, where it always has the same value on all possible solutions; thus, determining the corresponding cell is safe or not. The algorithm of the advanced reasoning strategy is shown as Algorithm 3.

Frontier division:. Suppose that there exists unsolved blocks $U_1, \ldots, U_k$ ($k \in \mathbb{N}^*$) on the board, and the frontiers of these unsolved blocks are $F_i = F(U_i)$ ($i = 1,\ldots,k$). Then, two different frontier F_1 and F_2 can be defined, where there exists an edge between F_1 and F_2 if $F_1 \cap F_2 \neq \phi$, and F_1 is dependent to F_2 if there exists edges that can connect F_1 and F_2. Moreover, for three different frontiers F_1, F_2, and F_2, if F_1 is dependent to F_2 and F_2 is dependent to F_3, then F_1 is dependent to F_3. Thus, hidden cells are divided into different independent frontiers $\hat{F}_1, \ldots, \hat{F}_n$. Fig. 5(a) is an example of Frontier Division where the blue, orange and green parts are independent (i.e., The distribution of mines in one part would not affect other parts).

Boolean modeling:. Suppose that $\hat{F} = \{c_1, \ldots, c_k\}$ is an independent frontier, for any cell $c_i \in \hat{F}$, let $W(c_i) = \{c^* | c^* \in N(c_i), 1 \leq I(c^*) \leq 8\}$, where $W(c_i)$ is a set of number cells that are in the neighborhood of c_i. And, let $W = W(c_1) \cup \cdots \cup W(c_k)$, where W is a union of all the $W(c_i)$. Then, each $c_i \in \hat{F}$ is a variable $x_i \in \{0,1\}$, where "0" means c_i is a safe cell, and "1" means c_i is a mine cell. Then, a system of linear equations can be obtained according to $W = \{w_1, \ldots, w_q\}$. The steps are as follows:

- **Step 1**: Initialize E and V as two empty lists.
- **Step 2**: For $i = 1,\ldots,k$, let $V[i] = c_i$, i.e., c_i is the ith element in V.
- **Step 3**: For $j = 1,\ldots,q$, establish an equation and let it be $E(j)$.

Each equation of E has the form as:

$$a_1 x_1 + \cdots + a_k x_k = b \tag{2}$$

where $i = 1,\ldots,k$, $a_i \in \{0,1\}$, $x_i \in \{0,1\}$, $b \in \{1,2,3,4,5,6,7,8\}$. V can be seen as a set of all bijective mappings between c_i and x_i. A mapping is bijective if and only if it is injective and surjective. For any $w \in W$, the left side of the equation is the sum of $V[\{c|c \in N(c), I(c) = 9\}]$, the right side is $I(w) - |\{c|c \in N(c), I(c) = 10\}|$. Fig. 5(b) is an example of modeling the hidden cells (blue color) into Boolean variables. We can see $(4,1)$ as x_1, $(4,2)$ as $x_2,\ldots,(4,6)$ as x_6, then $V[1] = (4,1), \ldots, V[6] = (4,6)$. The system of linear equations is shown as (3), where $x_1,\ldots,x_6 \in \{0,1\}$.

$$\begin{cases} x_1 + x_2 & = 1 \\ x_1 + x_2 + x_3 & = 2 \\ x_2 + x_3 + x_4 & = 2 \\ x_3 + x_4 + x_5 & = 2 \\ x_4 + x_5 + x_6 & = 2 \end{cases} \tag{3}$$

Gauss–Jordan Elimination algorithm:.

Definition 11. For a system of linear equations $\Phi : \boldsymbol{Ax} = \boldsymbol{b}$, a variable x is an invariable of Φ if x always equals the same value in all possible solutions of Φ.

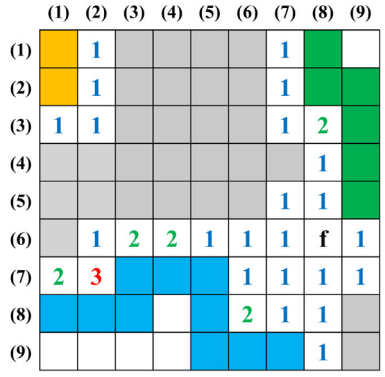
After modeling hidden cells in an independent frontier to Boolean variables, a system of linear equations can be established as (4):

$$\Phi : \boldsymbol{Ax} = \boldsymbol{b} \tag{4}$$

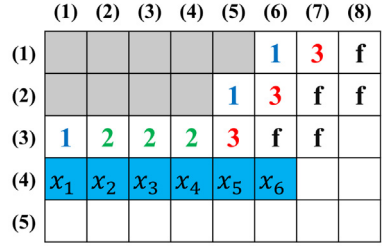
where for any element in $\boldsymbol{A}$ or in $\boldsymbol{x}$, the value is 0 or 1. The problem is to find invariables of Φ. Here, the Gauss–Jordan elimination algorithm is used to solve the system of linear equations by transforming the augmented matrix of the linear system to its reduced row echelon form, which transforms the original linear system into a more straightforward equivalent system [24].

Elementary row operations are needed to find the reduced row echelon form of a matrix. Suppose that $\boldsymbol{B}[i]$ denotes the ith row of a matrix $\boldsymbol{B}$, there are three types of elementary row operations:

- **Row swapping**: Swap the two different rows $\boldsymbol{B}[i]$ and $\boldsymbol{B}[j]$, i.e., $\boldsymbol{B}[i] \leftrightarrow \boldsymbol{B}[j]$, $i \neq j$.
- **Row multiplication**: Multiply each element in $\boldsymbol{B}[i]$ by a non-zero constant c, i.e., $\boldsymbol{B}[i] \leftarrow c\boldsymbol{B}[i]$, $c \neq 0$.



(a)



(b)

Fig. 5. (a) Frontier division: an example of 9 × 9 Minesweeper configuration with the cells marked with blue, orange, and green colors to signify different and independent frontiers; (b) Boolean model of hidden cells: an example of a partial 9 × 9 Minesweeper configuration (top part) with each hidden cells were modeled with Boolean variable $x_i \in \{0, 1\}$, where "0" means c_i is a safe cell, and "1" means c_i is a mine cell.

- **Row addition**: Add a multiple of a row $\boldsymbol{B}[j]$ on another row $\boldsymbol{B}[i]$, i.e., $\boldsymbol{B}[i] \leftarrow \boldsymbol{B}[i] + c\boldsymbol{B}[j], c \neq 0$.

If a matrix $\boldsymbol{C}$ can be obtained from another matrix $\boldsymbol{B}$ by elementary row operations, then $\boldsymbol{B}$ and $\boldsymbol{C}$ are equivalent. The function $JS : \boldsymbol{B} \rightarrow \boldsymbol{C}$ is the function with Gauss–Jordan elimination, where $\boldsymbol{B}$ is the augmented matrix of a system of linear equations, $\boldsymbol{C}$ is the reduced row echelon form of $\boldsymbol{B}$. Let $\boldsymbol{B}$ be the augmented matrix of Φ, and $\boldsymbol{C} = JS(\boldsymbol{B})$, i.e., $\boldsymbol{C}$ is the reduced row echelon form of $\boldsymbol{B}$. Suppose that $\boldsymbol{B}$ and $\boldsymbol{C}$ are $\alpha \times \beta$ matrices. For any $i \in \{1, \ldots, \alpha\}$ and $j \in \{1, \ldots, \beta\}$, $\boldsymbol{B}[i][j]$ denotes the element of the ith row and jth column of $\boldsymbol{B}$.

Conjecture 1. *For the ith row $\boldsymbol{C}[i]$, if there exists $k \in \{1, \ldots, \beta-1\}$ such that $\boldsymbol{C}[i][k] = 1$ and $\boldsymbol{C}[i][j] = 0$ for any $j \in \{1, \ldots, k-1, k+1, \ldots \beta-1\}$, then the variable $x_k = \boldsymbol{C}[i][\beta]$.*

Remark 1. Since $\boldsymbol{C}$ is the reduced row echelon form of $\boldsymbol{B}$, where $\boldsymbol{B}$ is the augmented matrix of a system of linear equations, $\boldsymbol{C}[i][j]$ $(1 \leq j \leq \beta - 1)$ is the coefficient of the variable x_j in the ith equation, thus, if only one element $\boldsymbol{C}[i][k] = 1$ in the first $\beta - 1$ elements, $x_k = \boldsymbol{C}[i][\beta]$.

An example of Conjecture 1: For the 2th row $\boldsymbol{C}[2]$, $x_1 + 0x_2 + 0x_3 = 1$, if $k = 1$, then $\boldsymbol{C}[2][1] = 1$, $\boldsymbol{C}[2][2] = 0$, and $\boldsymbol{C}[2][3] = 0$.

Conjecture 2. *For the ith row $\boldsymbol{C}[i]$, suppose that $\delta = \{j|j = 1, \ldots, \beta - 1, \boldsymbol{C}[i][j] \neq 0\}$, if $\boldsymbol{C}[i][\beta] = 0$, and $\boldsymbol{C}[i][k] > 0$ for any $k \in \delta$ or $\boldsymbol{C}[i][k] < 0$ for any $k \in \delta$, then, $x_k = 0$.*

Remark 2. Suppose that $\Delta = \{j|j = 1, \ldots, \beta - 1, \boldsymbol{C}[i][j] \neq 0\} = \{\delta_1, \ldots, \delta_t\}$. Since $x_{\delta_1}, \ldots, x_{\delta_t} \in \{0, 1\}$, if $\boldsymbol{C}[i][\delta_1] > 0, \ldots, \boldsymbol{C}[i][\delta_t] > 0$, then

$$\boldsymbol{C}[i][\delta_1]x_{\delta_1} + \cdots + \boldsymbol{C}[i][\delta_t]x_{\delta_t} \geq 0 \tag{5}$$

where equality $x_{\delta_1} = \cdots = x_{\delta_t} = 0$. (The case that $\boldsymbol{C}[i][\delta_1] < 0, \ldots, \boldsymbol{C}[i][\delta_t] < 0$ is similar, only need to replace "≥" with "≤"). Since $\boldsymbol{C}[i][\beta] = 0$, the equation is:

$$\boldsymbol{C}[i][\delta_1]x_{\delta_1} + \cdots + \boldsymbol{C}[i][\delta_t]x_{\delta_t} = 0 \tag{6}$$

From (5) and (6), $x_{\delta_1} = \cdots = x_{\delta_t} = 0$ is obtained.

Two examples of Conjecture 2 are shown as: $x_1 + x_2 + x_3 = 0$, and $-x_1 - x_2 - x_3 = 0$. The meaning is when the coefficient of variables are all positive or negative, then, $x_k = 0$.

Conjecture 3. *For the ith row $\boldsymbol{C}[i]$, suppose that $\Delta = \{j|j = 1, \ldots, \beta - 1, \boldsymbol{C}[i][j] > 0\}$, $\Gamma = \{j|j = 1, \ldots, \beta - 1, \boldsymbol{C}[i][j] < 0\}$. If $\boldsymbol{C}[i][\beta] = \sum_{k\in\Delta} \boldsymbol{C}[i][k]$, then for any $\delta \in \Delta$, $x_\delta = 1$, and for any $\gamma \in \Gamma$, $x_\gamma = 0$. If $\boldsymbol{C}[i][\beta] = -\sum_{k\in\Gamma} \boldsymbol{C}[i][k]$, then for any $\gamma \in \Gamma$, $x_\gamma = 1$, for any $\delta \in \Delta$, $x_\delta = 0$.*

Remark 3. Suppose that $\Delta = \{j|j = 1, \ldots, \beta - 1, \boldsymbol{C}[i][j] > 0\} = \{\delta_1, \ldots, \delta_t\}$, $\Gamma = \{j|j = 1, \ldots, \beta - 1, \boldsymbol{C}[i][j] < 0\} = \{\gamma_1, \ldots, \gamma_h\}$. Since $x_{\delta_1}, \ldots, x_{\delta_t} \in \{0, 1\}$, with equality $x_{\delta_1} = \cdots = x_{\delta_t} = 1$. Since $x_{\gamma_1}, \ldots, x_{\gamma_h} \in \{0, 1\}$, with equality $x_{\gamma_1} = \cdots = x_{\gamma_h} = 0$.

$$\boldsymbol{C}[i][\delta_1]x_{\delta_1} + \cdots + \boldsymbol{C}[i][\delta_t]x_{\delta_t} \leq \sum_{k=1}^{t} \boldsymbol{C}[i][\delta_k] \tag{7}$$

$$\boldsymbol{C}[i][\gamma_1]x_{\gamma_1} + \cdots + \boldsymbol{C}[i][\gamma_h]x_{\gamma_h} \leq 0 \tag{8}$$

From (7) and (8), Eq. (9) is obtained. Otherwise, since $\boldsymbol{C}[i][\beta] = \sum_{k=1}^{t} \boldsymbol{C}[i][\delta_k]$, the equation is shown as (10):

$$\boldsymbol{C}[i][1]x_1 + \cdots + \boldsymbol{C}[i][\beta - 1]x_{\beta-1} \leq \sum_{k=1}^{t} \boldsymbol{C}[i][\delta_k] \tag{9}$$

$$\boldsymbol{C}[i][1]x_1 + \cdots + \boldsymbol{C}[i][\beta - 1]x_{\beta-1} = \sum_{k=1}^{t} \boldsymbol{C}[i][\delta_k] \tag{10}$$

From (9) and (10), $x_{\delta_1} = \cdots = x_{\delta_t} = 1$ and $x_{\gamma_1} = \cdots = x_{\gamma_h} = 0$.

Here, δ and Γ represent the set of positive and negative coefficients, respectively. Two examples of Conjecture 3 are shown as: $x_1 + x_2 - x_3 = 2$, and $-x_1 - x_2 + x_3 = -2$. The meaning is when the coefficient of variables are all positive or negative, then, $x_k = 0$.

3.1.3. First action strategy

The first action strategy aims to find the best position of the first opened cell and acquire more information that determines the safety of the following cells. In other words, choosing the best initial position can significantly improve the winning rate. A study by Von Neumann [25] defined the neighborhood as a set of cells relative to the first cell, while Becerra [26] supposed that none of the first cell's neighbors are mines. Here, the safe neighborhood rule is proposed with the basis of the first action rule and the neighbor cell, and both are for the first cell opened. The discussion of the safe first action rule and the safe neighborhood rule is as follows.

Safe first action rule:. The safe first action rule is that the first opened cell must be safe [26]. Suppose that c_0 is the first opened cell, and mines will be placed in other cells $\{c|c \neq c_0\}$. Then, assume that the game configuration is $R \times L|b$, and $RL - 8 \geq b$, the probability $Pr(I(c_0) = 0)$ is given by (11).

$$Pr(I(c_0) = 0) = \frac{\binom{RL-|N(c_0)|-1}{b}}{\binom{RL-1}{b}} \tag{11}$$

$$= \frac{(RL - |N(c_0)| - b)...(RL - |N(c_0)| - 1)}{(RL - b)...(RL - 1)} \tag{12}$$

Algorithm 3 Advanced Reasoning Strategy

Input B: game board
Output S_1: the set of safe cells, S_2: the set of mine cells, V: the bijective mappings, C: the row echelon form

Function ARS(B)
  $S_1, S_2 \leftarrow \{\}$ ▷ empty set
  Do frontier division to get independent frontiers
  for each independent frontier $\hat{F}$ **do**
    Switch cells in $\hat{F}$ to variables
    Establish a system of linear equations Φ
    $V \leftarrow$ bijective mappings between cells and variables
    Do Gauss-Jordan elimination to Φ
    $C \leftarrow$ the row echelon form of the augmented matrix of Φ
    for $i = 1, ..., \alpha$ **do**
      $\Delta \leftarrow \{j | j = 1, ..., \beta - 1, \boldsymbol{C}[i][j] > 0\}$
      $\Gamma \leftarrow \{j | j = 1, ..., \beta - 1, \boldsymbol{C}[i][j] < 0\}$
      if $\Delta = \phi$ AND $\Gamma = \phi$ **then**
        Go to next loop
      end if
      if $|\Delta \cup \Gamma| = 1$ **then**
        if $\boldsymbol{C}[i][\beta] = 0$ **then**
          Add $V[j](j \in \Delta \cup \Gamma)$ into S_1
        else
          Add $V[j](j \in \Delta \cup \Gamma)$ into S_2
        end if
      else if $\boldsymbol{C}[i][\beta] = 0$ AND $(\Gamma = \phi$ OR $\Delta = \phi)$ **then**
        Add $V[j](j \in \Delta \cup \Gamma)$ into S_1
      else if $\boldsymbol{C}[i][\beta] = \sum_{k \in \Delta} \boldsymbol{C}[i][k]$ **then**
        Add $V[j](j \in \Gamma)$ into S_1
        Add $V[j](j \in \Delta)$ into S_2
      else if $\boldsymbol{C}[i][\beta] = -\sum_{k \in \Gamma} \boldsymbol{C}[i][k]$ **then**
        Add $V[j](j \in \Delta)$ into S_1
        Add $V[j](j \in \Gamma)$ into S_2
      end if
    end for
  end for
  Return S_1, S_2, C, V
end Function

To get the best first action, the first opened cell c_0 that maximizes the probability to "0" is need to be found. The cardinality of c_0 is defined as (13). Since $R - 8 \geq b$, $|N(c_0)| = 3$ can maximize $Pr(I(c_0) = 0)$. Thus, the corner cells are the best first action under the safe first action rule for $\frac{1}{4}$ of the board size.

$$|N(c_0)| = \begin{cases} 3, & c_0 \text{ is a corner cell} \\ 5, & c_0 \text{ is a border cell} \\ 8, & \text{other cases} \end{cases} \tag{13}$$

Safe neighborhood rule:. The first action strategy explores the safe neighborhood rule to enhance the winning rate, which means the first opened cell and its neighborhood must be safe. Suppose that c_0 is the first opened cell, and mines would be placed in other cells $\{c | c \neq c_0, c \notin N(c_0)\}$. In this case, the probability c_0 is given by (14).

$$Pr(I(c_0) = 0) = 1 \tag{14}$$

Such conditions allow for obtaining more information when c_0 was found. Suppose that F is the frontier after the first action, and $Z = \{c | c \in F, I(c) = 0\}$. The problem switches to maximizing $|Z|$ by opening the first cell. $|Z|$ contains the cells in the frontier after the first cell is opened, and the more the number of cells, the more information is obtained. This implies that the closer c_0 to the center of the board is, the larger the value of $|Z|$.

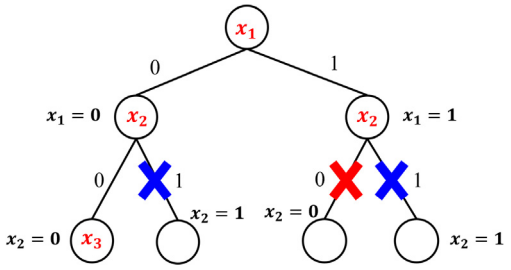


Fig. 6. Binary tree search for free variables: x_1 and x_2 are free variables, the nodes in the third rows are leaf nodes, which represent four variable states: $x_1 = 0, x_2 = 0$; $x_1 = 0, x_2 = 1$; $x_1 = 1, x_2 = 0$; and $x_1 = 1, x_2 = 1$.

3.1.4. Guessing strategy

The guessing strategy extends the two preceding reasoning strategies, which corresponds to the stochastic elements during the solving process of Minesweeper, where each cell on the board has uncertainty, and the player cannot determine the safe cell from the available information. However, players of different abilities use the guessing strategy differently; for example, advanced players avoid using the guessing strategy whenever possible, while beginners may use this strategy all the time. The key of the guessing strategy is to find a hidden cell that maximizes its safety probability. The mathematical solving logic of expert players is inspired by the skill of computing possibilities based on limited information. After dividing variables into free and leading variables by Gauss–Jordan elimination, the binary tree search searches all possible combinations of free variables to find numerical solutions. Finally, to reduce complexity, the pruned binary tree was obtained according to the constraints on the general solutions of linear systems.

Constrained search:. Suppose that there is one independent frontier on the board, and the advanced strategy is used to get a system of linear equations Φ. Let $\boldsymbol{B}$ be the augmented matrix of Φ, where $\boldsymbol{B}$ is a $\alpha \times \beta$ matrix. $\boldsymbol{C}$ is the reduced row echelon form of $\boldsymbol{B}$, which is got by Gauss–Jordan elimination [27]. The purpose is to find a hidden cell c with minimal risk and would provide more information. The Gauss–Jordan elimination divided variables into free variables V_f and leading variables V_l. Any variable $v_l \in V_l$ is a linear combination of free variables in V_f and a constant, so the solutions are presented by free variables.

Since the value of all variables is in $\{0, 1\}$, the binary tree search is used to find all possible numerical solutions of Φ, where the critical point is needed to search free variables that can significantly reduce the scale of the search. The binary tree has f free variables, and the depth of the root node is 0. The nodes with the depth less than f are non-leaf nodes, which variable equals the depth $d(1 \leq d \leq f - 1)$. The left/right child node of a non-leaf node n, and the variable is 0/1. Fig. 6 shows an illustration of binary tree search for free variables.

The definitions of the illegal node and the completed node are as follows.

Definition 12 (*Illegal Node*)**.** Suppose that the remaining mines on the board are r. A node n is an illegal node if either of two conditions is satisfied:

- The number of free variables with the value of 1 is greater than r.
- There exists a leading variable that is not in $\{0, 1\}$ under the variable state of n.

Definition 13 (*Completed Node*)**.** A node n is a completed node if either of two conditions is satisfied:

Table 1
Comparison for first action rule and safe first action rule based on different Minesweeper configuration.

Rule	X	Y	Z
First Action Rule	77%	66.2%	31.6%
Safe First Action Rule	87.9%	78.2%	39.7%

X: $9 \times 9|10$ mines; Y: $16 \times 16|40$ mines; Z: $16 \times 30|99$ mines.

- n is a leaf node.
- Each child node of n is a completed node or an illegal node.

By Definitions 12 and 13, the binary tree search can be pruned. For example, if a node is an illegal node, it is no need to search its child nodes. Otherwise, instead of storing the whole tree when searching the tree, it only needs to store the states of nodes from the root node to the current node, and each legal leaf node presents a possible free variable. Thus, all possible numerical solutions of Φ are obtained. Fig. 6 shows an example of tree pruning. The remaining mine on the board is $r = 1$, a leading variable $x = x_2 - x_1$, if $x_2 = 1$, $x_1 = 0$, then $x = 1$, the number of 1 would be two (greater than r), thus it should be pruned. If $x_2 = 0$, $x_1 = 1$, then $x = -1 \notin \{0, 1\}$, thus it should be pruned. If $x_2 = 1$, $x_1 = 1$, there would be two 1 (greater than r), thus it should be pruned. When creating a new node n, it should be pruned if n does not satisfy the constraints.

Additional enhancements:. The additional technique is to choose a variable that would provide more information when the values of variables $\hat{x_1}, \ldots, \hat{x_t}$ are near to "0".

Definition 14 (*Invariable*)**.** For an overdetermined system of equations, if some of the variables are fixed values, then we call them invariables.

Then, it is assumed that $x_i = 0$ for $i = 1, \ldots, t$, substitute $x = 0$ back into Φ by backtracking method, and denote $Inf(x_i)$ as the number of invariables of $\bar{\Phi}$. Finally, the x_i is chose with the largest $Inf(x_i)$. However, all possibilities of hidden cells are searched by brute-force search when the number of hidden cells on the board is insufficient. Since there is no information on the board, the best choice is to open a cell far from the solved area since it has a higher probability of opening a "0" cell. The algorithm of the guessing strategy is as Algorithm 4.

Algorithm 4 Guessing Strategy

Input B: game board, S: the set of solutions of linear systems
Output c_0: the cell that will be opened

```
Function GS(B)
    if S = φ then
        Choose a hidden cell c_0 on B far from the solved areas randomly
    else if the number of hidden cells on B is less than 6 then
        Do a brute-force search to choose the safest c_0 on B
    else
        Cand ← {}
        for each s ∈ S do
            V_f ← a list of free variables
            Create the root node n_r
            n_c ← n_r                                   ▷ current node
            P ← {}
            while n_r is not a completed node do        ▷ Search free variables
                if n_c is a leaf node then
                    Substitute the value of the free variables into s
                    Add a possible numerical solution into P
                    n_c ← the parent node of n_c
                    Update the state of n_c
                else if n_c is a completed node OR n_c is illegal then
                    n_c ← the parent node of n_c
                    Update the state of n_c
                else if the left child node of n_c was not created then
                    Create a left child node n_0
                    n_c ← n_0
                else
                    Create a right child node n_1
                    n_c ← n_1
                end if
            end while
            Calculate probabilities that cells in p are safe
            Add the safest cells and their probabilities into Cand
        end for
        c_0 ← the safest cell in Cand
    end if
    Return c_0
end Function
```

4. Experiment results and analysis

4.1. First action rule and safe first action rule

The first action rule is used for the first step on the board of one instance, which is uncertain that the first cell opened is safe, and the safe first action rule means the first opened cell is safe. This paper conducted experiments on both rules to simulate and compare the winning rate for each configuration. It is worth noting that the safe first action rule is working to increase the probability of the solving proceeding in case failing at the first step. In addition, the experiments are based on the "PAR" strategy, and the reason for not using "PAG" is to highlight the effectiveness of the "G" strategy later in the later experiments.

The experiment results under the first action rule and safe first action rule are showed in Table 1. The AI agent proposed based on the "PAR" strategy was running for 10,000 games for the Minesweeper configuration of X, Y, and Z. It can be observed that the winning rate of the safe first action rule achieves 87.9%, 78.2%, 39.7%, which is 10.9%, 12%, 8.1% higher than the first action rule.

4.2. Safe first action rule and safe neighborhood rule

The experimental simulations of the safe first action rule and the safe neighborhood rule are also for the first step on the board. The experiment focuses on comparing the effect of the location of the first cell being opened on the winning under the two rules. At the first step of solving Minesweeper, more safe cells opened means more information can be obtained. Thus, it is better to use other strategies to get more safe cells. Moreover, that is the basic idea of the safe neighborhood rule, and the specific rule is that the neighbor cells around the first opened cell are safe. The difference between the safe first action rule and the safe neighborhood rule is the different location of the first action cell.

Fig. 7 shows the winning rate of choosing different first opened cells under the safe first action rule and the safe neighborhood rules based on the game configuration of $8 \times 10|12$. Figs. (a) and (b) both show a quarter of the whole board. To better demonstrate the results under these rules, this experiment takes Minesweeper in configuration $8 \times 10|12$ as an example and 10,000

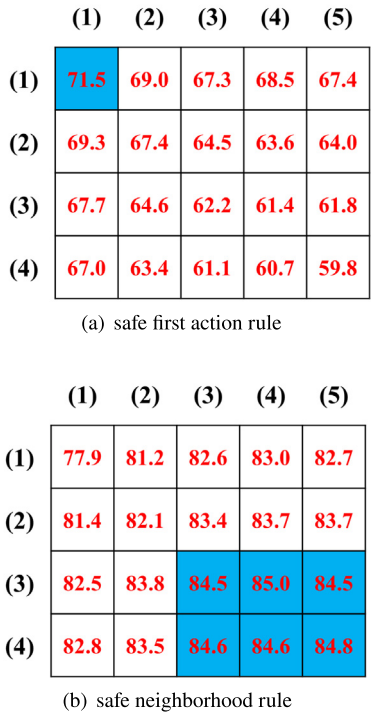


Fig. 7. (a) The winning rate of different first opened cells (game configuration: $8 \times 10|12$ mines with "PAR" AI agent strategy on safe first action rule). (b) The winning rate of different first opened cells (game configuration: $8 \times 10|12$ mines with "PAR" AI agent strategy on safe neighborhood rule).

runs per cell as the first open cell for both the safe first action rule and the safe neighborhood rule.

The AI agent is based on the "PAR" strategy, where "R" is the random guessing strategy. The result shows that the cell on the top left corner has the highest winning rate of 71.5% based on the safe first action rule, while the cell closer to the center with the winning rate around 85% under the safe neighborhood rule. Furthermore, it can obtain from figure (b) that the cell closer to the center has a higher winning rate, which suggests that the AI solver following the safe neighborhood rule has a better chance of getting a high winning rate. Since the correctness of the safe neighborhood rule has been proven by the experimental results above, all the simulation experiments utilizing the PFAG strategy are based on the safe neighborhood rule.

4.3. Comparison of methods and strategy

The difference between PAFG and PAFR strategy is that "G" and "R" are different, where "R" randomly guessing cells to open the cells, while "G" chooses the most promising safe one. The game configurations of X, Y, and Z are features of Microsoft Windows. In order to compare the experimental data with the previous studies, the experiment was conducted 10,000 times with three configurations X, Y, Z listed in Table 2. Compared with the PAFR strategy, the PAFG strategy improved the winning rate for X, Y, and Z by 2.6%, 6.7%, and 18.1%, respectively.

By comparing the first action strategy ("F") and the guessing strategy ("G"), the above results in Tables 1 and 2, and Fig. 7 show that the winning rate increases with these strategies, and securing the first cell and the neighbors has a significant impact for all the beginner, intermediate and expert configurations of Minesweeper. Moreover, the relationship between these four strategies is progressive, the "F" strategy is for the first opened cell on the board, and the following strategies are for the least possible use of more advanced strategies, in other words, solving the problem with as few strategies as possible to reduce the complexity of the algorithm, and the four strategies are in a progressive relationship. i.e., when using the "G" strategy, the other three strategies are not working.

Table 2
Comparison of various methods and strategies from the previous works against the proposed PAFG strategy performance (winning rate in percentage) in solving Minesweeper.

Strategy	Beginner	Intermediate	Expert
PAFG* (this paper)	96.4%	86.3%	45.6%
PAFR* (this paper)	93.8%	79.6%	27.5%
PAFG (this paper)	82%	86.3%	45.6%
PSEQ-D256 [15]	81.8%	78.2%	40.1%
PSEQ [15]	81.6%	78.1%	39.6%
OH [14]	80.2%	74.4%	38.7%
cSimEnuLoClf [13]	80.0%	75.6%	37.5%
LSWPE [12]	N.A.	67.7%	25.0%
Lordeiro (ϵ-greedy) [16]	77%	57.9%	4.1%
Lordeiro (UCB) [16]	76.4%	44.5%	0.62%
CSPStrategy† [11]	80.0%	44.3%	33.9%
Pedersen◇ [12]	92.5%	67.7%	25%

beginner: $8 \times 8|10$, intermediate: Y, and expert: Z;
*: beginner: X, intermediate: Y, and expert: Z;
†: beginner: $8 \times 8|10$ mines, intermediate: $15 \times 13|40$ mines, and expert: Z;
◇: beginner: $10 \times 10|10$ mines, intermediate: Y, and expert: Z.

To the best of our knowledge, the strongest AI in previous works is PSEQ-D256 [15]. However, the proposed strategy has higher winning rates of 0.2%, 8.1%, and 5.5%, for beginners, intermediate, and experts; thus, the performance is generally better than PSEQ strategies. The difference between PSEQ-D256 and PSEQ is that it uses several situations to choose OPTIMAL procedure or HEURISTIC strategies. Moreover, it is a solver composed of heuristic strategy and the quasi-optimal procedure while using the criterion of the number of situations; that is also why the performance is different from the proposed strategy in this paper.

It is worth noting that Tu et al. [15] had probed the corner blocks to be the first step to improving their strategies' success rate. This condition is well validated by this paper's safe first action rule. Moreover, this study proposed the safe neighborhood rule to enhance the performance of the PAFG strategy. As a result, even with the random guessing strategy (R), the winning rate for beginner and intermediate shown in Table 2 is higher than guessing strategies "P" and "T" in PSEQ. After that, Tu et al. [15] discussed the PSEQ strategies based on the maximum probability of the block and heuristic methods; this optimal and heuristic-based strategy has a much higher win rate than the previous strategy. However, a remote similarity between the "G" strategy in PAFG and PSEQ maximizes the probability of the hidden safe cell. At the same time, the constrained search and Gauss–Jordan elimination was used to reduce the scale of the search, together with the backtracking method to enhance the strategy. Thus, even without using heuristic methods, PAFG provides a high-level solver of Minesweeper compared with the solvers state of the art. Nevertheless, the difficulty of solving Minesweeper increases as the density of mines increases, which means the "G" strategy needs to be used more often and choose the safest cell from the cell with currently available information, without considering the part of the cell with unknown information.

5. Discussion

This study focuses on classic Microsoft Minesweeper by proposing an AI solver based on the PAFG strategy that reaches the reported winning rate (see Table 2) to facilitate the study of the puzzle-solving process. The four strategies proposed were

adopted to imitate different player abilities in solving problems, specifically, addressing a single-agent game, such as Minesweeper. With the presence of stochastic elements (e.g., hidden information of the mine location from all available cells on the board), the solvability is dependent on improving the condition that can be regarded as a 'puzzle' while mitigating the condition that can be regarded as a 'game'. In another word, mitigating the most on the ratio of certainty against uncertainty.

Moreover, the definition of solvability that distinguishes 'puzzle' and 'game' from the context of the stochastic single-agent game had been introduced and expanded from the two-player zero-sum game. By adopting an appropriate Minesweeper AI strategy, the experiment conducted on the various board size achieved adequate and comparable performance to the strategy of the related works. With the reported winning rate of the proposed PAFG given in Table 2, it is clear that based on the configuration of the experiments, for instance, the 9×9 Minesweeper was solvable 82% of the time as a puzzle. Meanwhile, Minesweeper was played as a game for the remaining 18% of the time. This condition can be associated with the information dynamics of the solving process where the random/hidden information of the single-agent game affected the solvability of the positions/moves, which makes it either solvable or not.

From the solvability point of view, the strategy of Minesweeper AI in solving problems is based on cells with information on the board, since in most cases, the safety probability of cells with information is more distinctive, which decides to be more informed compared to the cells without any information. Moreover, in some cases, it is easy to determine whether the hidden cells on the board are mines or not. Such a situation is when there is a relationship between the number of hidden cells and the number of mines minus the number of flag cells, which constitutes the primary reasoning strategy ("P" strategy). However, the purpose is to find all hidden mines on the board. As the information increases, the problem is switched to finding variables in a Boolean variable linear system of equations, which is over-determined in theory. Here, Gauss–Jordan elimination helped reduce the augmented matrix into the row echelon form. The general solutions to the system can be obtained with the advanced reasoning strategy ("A" strategy). Moreover, the guessing strategy ("G" strategy) was used to find a hidden cell and maximize its safe probability when all cells have a probability of being mines instead of open cells randomly. In addition, the rule proposed in the first action strategy ("F" strategy) always keeps both the first cell opened and the surrounding cells safe to obtain more information in the first step.

The strength of the PAFG strategies was the complementary nature of the four strategies to achieve a high winning rate in solving Minesweeper. In other words, each strategy provides a balance between what is currently known and what is unknown of the puzzle state to decide the best possible future state of the puzzle. Based on the reported experimental results, this condition showed that having such a balance is well-suited for Minesweeper, where stochastic elements are always present, inducing insufficient state of the art approaches in dealing with such cases.

Nevertheless, if there is no available information on the board, opening a cell far from the solved area is the best choice. The current maximum probability of safe cells was obtained using a conditional probability based on known information. However, there are cases where the maximum probability of safe cells based on known information is lower than that of cells in regions of unknown information, such as when the density of the remaining mines is low. Thus, balancing exploration and utilization based on known information is an expecting direction of future improvement.

6. Conclusions

This paper focused on defining the solvability of the single-agent stochastic puzzle (such as the classic Minesweeper), where the border between puzzles and games was considered based on the winning rate (p). For example, a deterministic puzzle is a puzzle with a winning rate of $p = 1$, which implies the puzzle is deterministic and strongly solved. On the other hand, a 'stochastic puzzle' is identified when $0 < p < 1$, which implies the game is stochastic and is weakly solved. Then, $p = 0$ is identified as a 'game,' which is stochastic and not solvable. Such definition was achieved by proposing an AI solver with a solving strategy, called the PAFG strategy, where "P" is the primary reasoning strategy, "A" is the advanced reasoning strategy, "F" is the first action strategy, and "G" is the guessing strategy.

A knowledge-based rule verified by the experimental simulation had found that cells closer to the center are the best positions for the "F" strategy with the safe neighborhood rule. Furthermore, other strategies were used to obtain a high winning rate based on available information alongside game progression. For example, the safety probability of each surrounding cell was independently calculated. In addition, the advanced reasoning strategy is used to invariably find a linear system of equations based on the primary reasoning strategy mathematically. Moreover, the Gauss–Jordan elimination is adopted to obtain general solutions in a linear system of equations, exploring the guessing strategy to independently reduce the augmented matrix into the row echelon form under the information of the considered cell. Compared with the solving strategy state of the art, this AI solver reached a high winning rate with: 96.4% for $9 \times 9|10$ mines, 86.3% for $16 \times 16|40$ mines, and 45.6% for $16 \times 30|99$ mines.

The design concept of this AI solver is based on the solving logic of players, which is modeled mathematically to facilitate future entertainment analysis for different abilities players to solve Minesweeper. Furthermore, more strategies correspond to the increased player's ability, which implies a higher winning rate, verified by proposed knowledge-driven reasoning, strategies, and simulation conducted in this study. Promising future work includes exploring the PAFG strategy in other related single-agent stochastic puzzles, such as 2048 and other similar puzzles, to expand further and verify the definition established from this study. Moreover, adopting the PAFG strategy to analyze and measure the entertainment aspects of the single-agent stochastic and deterministic puzzles is an exciting prospect for future investigation.

CRediT authorship contribution statement

Chang Liu: Conceptualization, Methodology, Software, Data curation, Investigation, Formal analysis, Writing – original draft, Investigation, Visualization, Writing – review & editing. **Shunqi Huang:** Conceptualization, Methodology, Investigation, Software. **Gao Naying:** Methodology, Software, Data curation, Formal analysis, Visualization. **Mohd Nor Akmal Khalid:** Visualization, Writing – review & editing, Project administration. **Hiroyuki Iida:** Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] L.V. Allis, et al., Searching for Solutions in Games and Artificial Intelligence, Ponsen & Looijen Wageningen, 1994.

[2] H. Kita, C. Alessandro, H. Iida, Theoretical value prediction in game-playing, SIG-GI, IPSJ (GI) 87 (2005) 71–77.

[3] H. Iida, On games and fairness, in: 12th Game Programming Workshop in Japan, 2007, pp. 17–22.

[4] A. Kehagias, G. Konstantinidis, Some game-theoretic remarks on two-player generalized cops and robbers games, Dynam. Games Appl. (2021) 1–18.

[5] G. Kendall, A. Parkes, K. Spoerer, A survey of NP-complete puzzles, ICGA J. 31 (1) (2008) 13–34.

[6] C. Browne, The nature of puzzles, Game Puzzle Des. 1 (1) (2015) 23–34.

[7] M.S. Kiarostami, M. Daneshvaramoli, S. Khalaj Monfared, A. Visuri, H. Karisani, S. Hosio, H. Khashehchi, E. Futuhi, D. Rahmati, S. Gorgin, On using Monte-Carlo tree search to solve puzzles, in: 2021 7th International Conference on Computer Technology Applications, 2021, pp. 18–26.

[8] A. Primanita, M.N.A. Khalid, H. Iida, Computing games: Bridging the gap between search and entertainment, IEEE Access PP (2021) 1, http://dx.doi.org/10.1109/ACCESS.2021.3079356.

[9] B. Bonet, H. Geffner, Belief tracking for planning with sensing: Width, complexity and approximations, J. Artificial Intelligence Res. 50 (2014) 923–970.

[10] W. Tingting, H. Shunqi, H.P.P. Aung, M.N.A. Khalid, H. Iida, Analysis of single-agent game: Case study using minesweeper, in: 2020 International Conference on Advanced Information Technologies, ICAIT, IEEE, 2020, pp. 105–110.

[11] C. Studholme, Minesweeper as a constraint satisfaction problem, Unpublished Project Report, 2000.

[12] K. Pedersen, The Complexity of Minesweeper and Strategies for Game Playing, Project Report, Univ. Warwick, 2004.

[13] M. Legendre, K. Hollard, O. Buffet, A. Dutech, MineSweeper: Where to Probe? (Ph.D. thesis), INRIA, 2012.

[14] S. Golan, Minesweeper strategy for one mine, Appl. Math. Comput. 232 (2014) 292–302.

[15] J. Tu, T. Li, S. Chen, C. Zu, Z. Gu, Exploring efficient strategies for minesweeper, in: Workshops at the Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[16] I. Lordeiro, D. Haddad, D. Cardoso, Multi-armed bandits for minesweeper: Profiting from exploration-exploitation synergy, IEEE Trans. Games PP (2021) 1, http://dx.doi.org/10.1109/TG.2021.3082909.

[17] K.M. Bayer, J. Snyder, B.Y. Choueiry, An interactive constraint-based approach to minesweeper, in: AAAI, 2006, pp. 1933–1934.

[18] A. Adamatzky, How cellular automaton plays minesweeper, Appl. Math. Comput. 85 (2–3) (1997) 127–137.

[19] R. Kaye, Minesweeper is NP-complete, Math. Intelligencer 22 (2) (2000) 9–15.

[20] S. Arora, B. Barak, Computational Complexity: A Modern Approach, Cambridge University Press, 2009.

[21] S.A. Cook, The complexity of theorem-proving procedures, in: Proceedings of the Third Annual ACM Symposium on Theory of Computing, 1971, pp. 151–158.

[22] A. Scott, U. Stege, I. Van Rooij, Minesweeper may not be NP-complete but is hard nonetheless, Math. Intelligencer 33 (4) (2011) 5–17.

[23] S. Langerman, Y. Uno, Threes!, Fives, 1024!, and 2048 are hard, Theoret. Comput. Sci. 748 (2018) 17–27.

[24] G. Strang, G. Strang, G. Strang, G. Strang, Introduction to Linear Algebra, vol. 3, Wellesley-Cambridge Press Wellesley, MA, 1993.

[25] J. Von Neumann, The General and Logical Theory of Automata, Wiley, 1951.

[26] D.J. Becerra, Algorithmic Approaches to Playing Minesweeper (Ph.D. thesis), 2015.

[27] G. Peters, J.H. Wilkinson, On the stability of Gauss-Jordan elimination with pivoting, Commun. ACM 18 (1) (1975) 20–24.