Proposal and Generation of Endgame Puzzles for an Imperfect Information Game Geister

Chu-Hsuan Hsueh[a,*], Takefumi Ishii[a], Tsuyoshi Hashimoto[b], Kokolo Ikeda[a,c]

[a] *Division of Advanced Science and Technology, Japan Advanced Institute of Science and Technology, Nomi, Ishikawa 923-1292 Japan*
[b] *Department of Information Engineering, National Institute of Technology, Matsue College, Matsue, Shimane, 690-8518, Japan*
[c] *Division of Transdisciplinary Sciences, Japan Advanced Institute of Science and Technology, Nomi, Ishikawa 923-1292 Japan*

Abstract

Geister is a two-player imperfect information game played with two kinds of pieces, blue and red, where each player cannot observe the colors of the opponent's pieces on the board. In this paper, we propose Geister endgame puzzles for players to enjoy the game in another form or to practice figuring out moves that are proven to win, similar to chess mating problems. In Geister endgame puzzles, the goal is to find the shortest winning moves under the assumption of the worst cases. We propose not-revealed and partly-revealed puzzles according to how the opponent's piece colors are revealed to the player. We also propose two kinds of special puzzles, capture-win and red-wall, that utilize specific victory conditions of Geister. We generate Geister endgame puzzles by randomly placing pieces on the board and then using a solver to check whether winning moves exist. The generation success rates for not-revealed and partly-revealed puzzles are approximately 20% to 30%. The experiments also show that puzzles with more moves to win are less frequently generated. In addition, we conduct preliminary experiments that invite beginners to evaluate generated puzzles, which shows that longer-win and special puzzles tend to be more difficult and interesting, respectively.

Keywords: Geister, imperfect information, endgame puzzle, content generation, entertainment

1. Introduction

Games have served as an important form of entertainment for humans for a long time. In the past decades, artificial intelligence (AI) in games has achieved

*Corresponding author
Email addresses: hsuehch@jaist.ac.jp (Chu-Hsuan Hsueh), s1810010@jaist.ac.jp (Takefumi Ishii), hashimoto@matsue-ct.jp (Tsuyoshi Hashimoto), kokolo@jaist.ac.jp (Kokolo Ikeda)

remarkable success, owing mainly to the rapid development of hardware and algorithms. A famous example is the AlphaZero algorithm [1], reaching superhuman levels of play in chess, shogi, and Go. Another widely studied field is employing AI to generate game content [2], among which puzzles are often used by human players for brain training or improving playing skills. For example, chess problems [3, 4, 5, 6] or life-and-death problems for Go [7] are popular among chess or Go players.

Compared to perfect information games such as chess, shogi, and Go, there are relatively few studies on content generation or entertainment in imperfect information games. Some famous imperfect information games are poker and Mahjong, where players cannot fully observe the opponents' information (e.g., the cards or tiles on their hands). To deal with this, players often need to make inferences from available information. Thus, developing algorithms for imperfect information games is considered more difficult than perfect information. Nevertheless, many real-world decision-making problems involve imperfect information. From these points, it is worth investigating imperfect information games, where the results are expected to provide insights into solving real-world problems.

In this work, we study an imperfect information game called Geister [8], which is a two-player board game with some characteristics in common with the world-famous chess and shogi. The game is played on a 6×6 board, and each of the two players owns eight pieces, four blue pieces and four red pieces, at the beginning. In the game, each player knows the colors (types) of their pieces but not the opponent's (similar to Stratego [9]). Despite its simple rules, many techniques are involved in the gameplay, such as inferring the opponent's piece colors, bluffing the opponent into misunderstanding the player's own piece colors, and playing moves that lead to doubtless wins when available. Thus, it may be a little challenging for Geister players to improve skills by directly playing games with other players. For example, regarding figuring out moves of doubtless wins, beginners sometimes overlook chances to win games where they could because they consider the situation too complicated.

In this paper, we propose *Geister endgame puzzles* for players to enjoy Geister in a different way and to help players learn techniques to be used in real play, similar to the mating problems of chess and shogi. In more detail, we propose two categories of Geister endgame puzzles: *not-revealed* and *partly-revealed*. The former follows the game rules that players cannot observe the opponents' piece colors. The latter, though different from game rules, is introduced considering that experienced players can usually infer some of the opponent's piece colors in endgames. We expect the strategies to play the two kinds of puzzles to be different, while both are valuable for training players. Moreover, we propose two kinds of special puzzles called *capture-win* and *red-wall* that utilize specific victory conditions of Geister, which are also important to consider during real play.

In this work, we generate Geister endgame puzzles by randomly placing given numbers of both players' pieces on the board and then filtering out non-playable ones by a solver based on the depth-first proof-number search [10]. In addition,

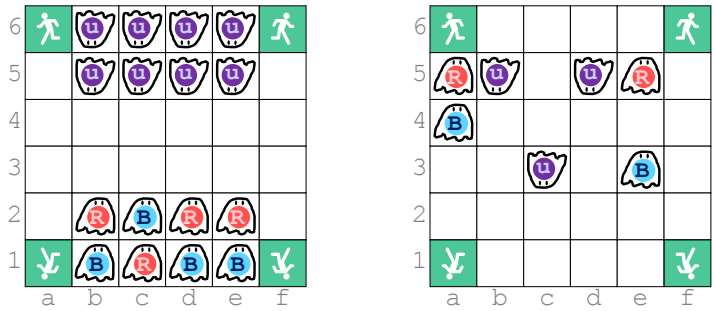


Figure 1: (a) An initial position and (b) an example position for Geister from the bottom-side player's view.

several techniques are proposed to speed up the solver. The experiments generate puzzles with at most 17 moves to win and with at most 3 pieces of each kind of both players' blue and red pieces. The random placement generation has success rates of 24.92% and 30.18% for not-revealed and partly-revealed puzzles, respectively, while the proposed techniques reduce 94.67% and 78.07% of computation time. The experiments also show that the success rates of generation decrease approximately exponentially as the number of moves to win increases.

The rest of this paper is structured as follows. Section 2 reviews background knowledge, including the game Geister and procedural puzzle generation. Section 3 defines Geister endgame puzzles, including two categories, not-revealed and partly-revealed. Section 4 introduces how puzzles are generated, and then Section 5 presents the results. Section 6 shows some examples of Geister endgame puzzles. Finally, Section 7 makes concluding remarks.

2. Background

This section introduces the game Geister and the current progress on Geister research in Subsection 2.1 and reviews procedural puzzle generation in Subsection 2.2.

2.1. Geister

First, Subsection 2.1.1 describes Geister's rules and some basic strategies. Subsection 2.1.2 then gives a brief review of Geister research.

2.1.1. Game Rules and Basic Strategies

Geister [8], also known as Ghosts, Fantasmi, and Phantoms-vs-Phantoms, is a two-player zero-sum deterministic imperfect information game played on a 6×6 board. Each player owns eight pieces, four for each of the two colors: blue (good ghosts) and red (evil ghosts). The most notable feature of the game is that the players cannot see the opponents' piece types, similar to Stratego [9]. At the

beginning of a game, both players can arrange their pieces in any permutation within the 2×4 areas in their two back rows. Fig. 1a shows an example of an initial arrangement from the bottom-side player's view. Uppercase letters `B` and `R` denote the bottom-side player's blue and red pieces, respectively. The lowercase letter `u` denotes the top-side player's pieces whose colors are unknown to the bottom-side player.

The two players move pieces alternatively. In each turn, players can move one of their pieces to an orthogonally adjacent square, either empty or with an opponent's piece. In the latter case, the opponent's piece is captured and removed from the game. When capturing the opponent's pieces, the player knows the captured pieces' colors (types). In other words, the players can always know the opponents' remaining numbers of red and blue pieces but do not know the exact locations.

A special rule for the game is related to *exits*, i.e., the four corners of the board (a1, f1, a6, and f6). More specifically, a player has two exits, which are the two corners in the farthest row from the player's view. Players can move their blue pieces to the outside of the board from the exits, which is called escape and costs one turn. For example, bottom-side players (starting at rows 1 and 2) can move their blue pieces at a6 and f6 to the outside of the board. Note that red pieces cannot escape from the board even when they are at the exit squares. The victory condition contains three rules: (1) a player wins by escaping one of the blue pieces, (2) a player wins by capturing all the opponent's blue pieces, and (3) a player wins when all the red pieces are captured by the opponent. The game rules did not define when games end as draws, but in practice, a maximum number of moves is set (e.g., 300 [11]).

The basic strategy to play Geister is to move blue pieces toward the exits while preventing the opponent's blue pieces from escaping. Without loss of generality, the following discussions are made from the bottom-side player's view. Despite such an easy-to-understand guideline, it is not wise to only move blue pieces to the exits at the top or capture whatever the top-side player's pieces move to the bottom-side exits. If such a strategy is known by the top-side player, for the former, the bottom-side player's blue pieces are likely to be captured before being able to escape. For the latter, the bottom-side player may lose by capturing all the top-side player's red pieces. From these points, players need to guess the opponents' piece colors and also try to bluff the opponents into misunderstanding the players' own piece colors. For the position in Fig. 1b, the bottom-side player may move the `R` at e5 to e6 or f5 to make it look like a blue piece (moving toward an exit and preventing it from being captured). The rules of Geister are simple, but sophisticated strategies are needed to win. Since the opponents' piece colors are unknown, psychological factors are involved, making it challenging for players to develop good strategies.

2.1.2. Research on Creating Strong AI Players

Most of the research on Geister focused on creating AI players. Researchers have tried various approaches, but the AI players still had difficulty reaching the

levels of average human players. One key aspect is the handling of the unknown information about the opponent's piece colors.

To deal with the uncertainty, some researchers tried to predict the opponent's piece colors. Aiolli and Palazzi [12] applied a simple machine learning algorithm to predict the piece colors from the game records of a *single* human player. They reported an accuracy of 74% on predicting the human player's piece colors. Farooq et al. [13] tried more machine learning algorithms training on the game records of *multiple* players who participated in Geister AI player competitions. The accuracy of predicting the AI players' piece colors was approximately 58%. They also employed Aiolli and Palazzi [12]'s algorithm on the same training data, where the accuracy was only approximately 52%. They suspected that different (AI) players had different strategies, making it challenging to make accurate predictions.

Some researchers further incorporated the predictions of piece colors to create AI players. Balakrishnan et al. [14] designed rule-based predictions on the opponent's piece colors. They used the prediction results to make the game become perfect information so that the minimax search could be applied. Mishio and Kotani [15] proposed an algorithm called *using past playout* (UPP) to evaluate states with different assumptions of the opponent's piece colors. They used the evaluation scores to assign probabilities of different assumptions to be selected in a Monte-Carlo method. Their experiments showed that UPP played better than the Monte-Carlo method which assigned equal probabilities to different assumptions. Buck et al. [16] used a neural network (NN) to predict the opponent's piece colors and designed a fuzzy inference system (FIS) to evaluate possible moves. They further developed a co-evolutionary algorithm to learn the parameters of the NN and the FIS. Their AI player won second place in the IEEE CIS Ghosts Challenge in 2013.

Another way to deal with the uncertainty was the *purple pieces* proposed by Sato [17]. In more detail, all the opponent's pieces were regarded as purple pieces, which always considered the worst cases and made the game perfect information. Namely, an opponent's piece was treated as red when captured and as blue when escaped. In addition to proposing the concept of purple pieces, Sato [17] also tried to search for winning moves and applied reinforcement learning to learn action-value functions based on self-play games. Kawakami and Hashimoto [11] listed possible configurations of the opponent's piece colors, making the game perfect information, and built one minimax tree for each. They analyzed several ways of utilizing these minimax search trees for move selection. In addition, they incorporated the concept of purple pieces into the minimax search and found the playing strength was greatly improved.

Chen and Kaneko [18] applied deep counterfactual regret minimization, a learning algorithm using deep NNs to approximate Nash equilibria. Their AI players had win rates over 75% against the random player who randomly selected one possible move. Tomoda and Hasebe [19] proposed to estimate hidden information using reinforcement learning, which learns from self-play games with perfect information. Their experiments showed that their approach performed better than directly learning from the imperfect information games.

2.2. Puzzle Generation

Puzzles play an important role for human players, not only from the entertainment aspect but also in helping players improve their skills. Procedural puzzle generation [20] aims to generate puzzles algorithmically, which is a branch of procedural content generation. This subsection reviews some approaches and applications.

Chess problems are a famous and classical puzzle genre. The automatic generation of chess problems has also been studied for a few decades. For example, Schlosser [3] built endgame databases by retrograde analysis and selected high-quality chess mating problems according to evaluation values. HaCohen-Kerner and Fainshtein [4] used a bounded depth-first search to improve the quality of existing chess problems designed by human composers. Iqbal [5] aimed to improve the aesthetics of chess mating problems from seven aspects, such as removing pieces and looking for a shorter mate. Bizjak and Guid [6] encoded tactical problems as text documents and automatically generated similar problems from collected chess games. Hirose et al. [21] generated tsume-shogi problems, i.e., shogi mating problems, by the *reverse method*. The algorithm searched in a backward way from checkmate positions to compose mating problems.

Puzzles in single-player games have also been widely investigated. Mantere and Koljonen [22] generated and rated Sudoku puzzles by a genetic algorithm (GA). Oranchak [23] applied GA to generate puzzles with desired qualities in another game called Shinro. Takahashi [24] targeted a game called Puyo-Puyo and aimed to train human players by automatically generated puzzles. In that work, two methods were compared, which were the random generation and the reverse method. The latter was found to be more efficient. Oikawa et al. [25] tried to improve human players' T-spin skills, an important technique in Tetris. They employed the reverse method to generate puzzles and used supervised learning to predict the interestingness and difficulty of puzzles. de Trenquelleon et al. [26] targeted the Rush Hour puzzles. They randomly generated puzzles and then employed neural networks to assess the playability and difficulty of the puzzles.

3. Geister Endgame Puzzles

Puzzles for board games have long been used both for entertainment and brain exercise. Especially, chess mating problems [3, 4, 5], tsume-shogi [21], and life-and-death problems for Go [7] are popular ones. When solving such puzzles, players need to imagine possible variations in mind, which can help them improve their playing skills. Also, one advantage is that these puzzles can be played by a single player, even for two-player games. Another advantage is that the time for solving one puzzle is usually shorter than playing one game from the beginning to the end.

Geister endgame puzzles proposed in this section aim to provide players with one more choice to practice the playing skills of Geister or to have another way of enjoying playing Geister. In this work, we assume endgame puzzles in

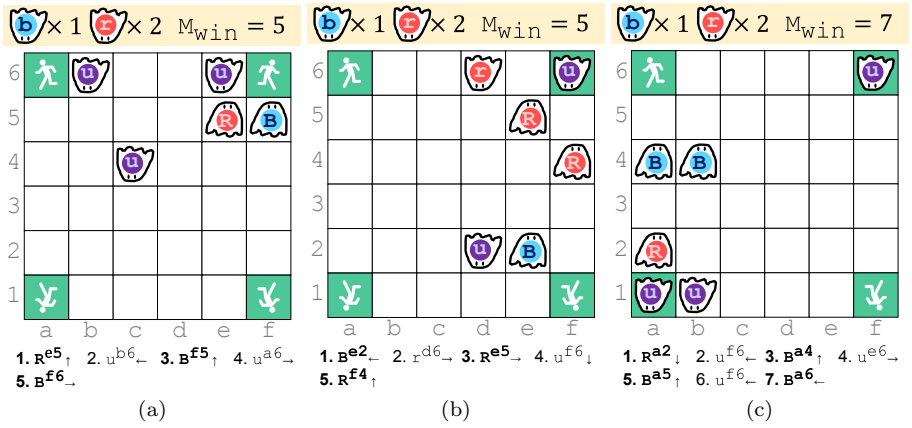


Figure 2: Examples of Geister endgame puzzles where it is the bottom-side player to move: (a) a not-revealed puzzle, (b) a partly-revealed puzzle won by capturing all the opponent's blue pieces, and (c) a not-revealed puzzle utilizing the player's red piece.

the imperfect information game Geister to have similar effects on training and entertaining players to perfect information games such as chess, shogi, and Go. In the following, Subsection 3.1 defines Geister endgame puzzles, Subsection 3.2 introduces two categories based on the information about the opponent's pieces, and Subsection 3.3 discusses Geister's three victory conditions and their roles in endgame puzzles.

3.1. Definition

A Geister endgame puzzle consists of the *position* and the *number of moves to win*, similar to chess mating problems and tsume-shogi.

- Without loss of generality, all puzzles are from the bottom-side player's view, and it is the bottom-side player's turn to move. The position contains information about piece arrangement and the numbers of the top-side player's blue and red pieces (N_b and N_r).
- The number of moves to win, M_{win}, counts both players' moves until the game ends, considering the worst cases from the bottom-side player's view and the optimal play between the two players. Under such assumptions, M_{win} is the least number of moves to win. Puzzles with M_{win} moves to win are referred to as M_{win}-move puzzles.

Other rules such as moving pieces and victory conditions are the same as the original Geister.

Even though Geister is an imperfect information game, if the bottom-side player is guaranteed to win in the worst case, the game is a doubtless win. By worst, the bottom-side player assumes that the top-side player's pieces are purple pieces (becoming blue pieces when escaped and red pieces when captured)

and that the top-side player knows the bottom-side player's piece colors. When the worst case is considered, the puzzles become perfect information. Fig. 2a shows an example of a Geister endgame puzzle where N_b, N_r, and M_{win} are 1, 2, and 5, respectively. The solutions of the puzzles in Fig. 2 will be explained in Subsection 3.3.

3.2. Categorization Based on Information about the Opponent's Piece Colors

According to Geister's game rules, the top-side player's piece colors should be unknown to the bottom-side player. However, in real play, experienced players can usually make some predictions according to past moves. Considering these facts, we propose two kinds of puzzles based on the information of the top-side player's piece colors, *not-revealed* and *partly-revealed*. Figs. 2a and 2c show two examples of not-revealed puzzles, and Fig. 2b shows an example of a partly-revealed puzzle. For partly-revealed puzzles, the lowercase letters `b` and `r` denote the top-side player's blue and red pieces, respectively.

3.3. Discussions on Three Victory Conditions

This subsection discusses the roles of Geister's three different victory conditions in endgame puzzles in Subsections 3.3.1 to 3.3.3.

3.3.1. Victory Condition 1: Escaping One of the Player's Blue Pieces

Escaping blue pieces is one of the most fundamental elements of Geister. Although creating trivial puzzles is easy, e.g., merely moving the blue piece closest to the exits for escaping, it is more desirable if the puzzles contain tactical situations. For example, the bottom-side player needs to prevent the top-side player's escapes first, or the bottom-side player needs to break through the top-side player's defense around the top-side exits. Puzzles with such tactical situations usually require more moves to win, which we consider closer to real play and useful for training players. For the puzzle in Fig. 2a, the solution is to capture the `u` at e6 by the `R` at e5 and then escape the `B` from f6. Since the worst cases assume that the captured `u` is a red piece instead of blue, the bottom-side player does not win immediately after capturing the `u` at e6. Also, if the bottom-side player moves the `B` to f6 first, the top-side player will win by capturing the bottom-side player's blue piece. Thus, M_{win} is 5 instead of other lower numbers.

3.3.2. Victory Condition 2: Capturing All the Opponent's Blue Pieces

One meaning of the worst cases in Geister endgame puzzles is that the top-side player's unknown pieces are regarded as red pieces when captured. Thus, for not-revealed puzzles, it is impossible for the bottom-side player to win by capturing all the top-side player's blue pieces. In other words, the bottom-side player will lose due to capturing all the top-side player's red pieces before capturing all blue pieces.

As for partly-revealed puzzles, as long as at least one of the top-side player's red pieces is revealed, it is possible for the bottom-side player to capture all the

top-side player's blue pieces. Fig. 2b shows an example where one of the top-side player's red pieces is revealed. The first move in the solution is to capture the `u` at d2 by the `B` at e2. Since the worst case assumes the `u` to be `r`, the bottom-side player needs to capture the remaining `u`, i.e., `b`, at f6 to win. In this work, we refer to such puzzles as winning by capture, or *capture-win*, and Subsection 4.3 will present how to identify capture-win puzzles.

3.3.3. Victory Condition 3: Having All Red Pieces Captured by the Opponent

Another meaning of the worst cases in Geister puzzles is to assume that the top-side player knows the bottom-side player's piece colors. In addition, under the assumption of optimal play, the top-side player tries to lose as late as possible. The top-side player can always avoid capturing the bottom-side player's last red piece. Thus, it is impossible for the bottom-side player to win by forcing the top-side player to capture all the bottom-side player's red pieces. However, this victory condition can be utilized to limit the top-side player's moves, which is an important technique in real play.

For the puzzle in Fig. 2c, the solution is to capture the `u` at a1 by the `R` at a2 and then move the `B` at a4 to a6 to escape. Capturing the `u` at a1 by `R` not only prevents the top-side player's piece from directly escaping but also protects the exit. Under the assumption of optimal play, the top-side player cannot capture the `R` trying to escape from a1, or the top-side player will lose immediately. Such a red piece is like a wall that blocks the top-side player's movements. In this work, we refer to puzzles that utilize this victory condition as *red-wall* puzzles, and Subsection 4.3 will present how to identify them.

4. Puzzle Generation Approach

As a preliminary step toward automatically generating Geister endgame puzzles, in this work, we randomly place pieces on the board, described in more detail in Subsection 4.1. Since it is not guaranteed that any piece arrangement becomes a Geister endgame puzzle, Subsection 4.2 introduces a solver to check the piece arrangements. Moreover, Subsection 4.3 explains how to identify capture-win and red-wall puzzles[1]. Readers who are less interested in the implementation details can skip this section, especially Subsections 4.2 and 4.3.

4.1. Random Placement Generation

The random placement generation starts with creating a Geister position by randomly placing a given number of pieces on the board. In more detail, the inputs to the method include four numbers for the bottom-side player's and the top-side player's blue and red pieces, denoted by (N_B, N_R, N_b, N_r). For example, (4, 1, 3, 2) means that the bottom-side player has 4 blue pieces and 1 red piece, while the top-side player has 3 blue pieces and 2 red pieces. All the

[1]The codes of this work are openly available at `https://github.com/hsuehch/Geister-Endgame-Puzzle`.

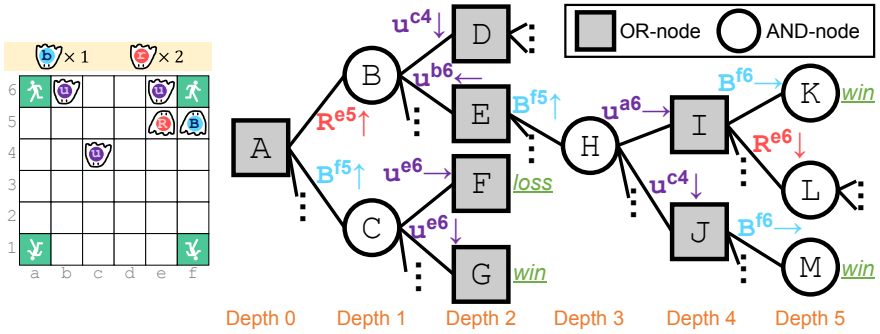


Figure 3: An AND-OR tree for the position in Fig. 2a, where moves are denoted by [piece][coordinate][direction], and wins and losses are from the bottom-side player's view.

pieces are randomly placed on the board, where no pieces share the same square. To avoid generating too-trivial puzzles, the bottom-side player's blue pieces are never placed on the top-side exits (a6 and f6). When generating not-revealed puzzles, all of the top-side player's pieces are not revealed to the bottom-side player, denoted by u (unknown). When generating partly-revealed puzzles, we randomly decide whether each top-side player's piece is revealed. However, to meet the criteria of partly revealed, at least one of the top-side player's pieces should be revealed. Also, at least one blue piece and one red piece should remain unknown to the bottom-side player.

We then employ the solver in Subsection 4.2 to check whether the bottom-side player can win from the position. If it is the bottom-side player's win, the solver also reports the number of moves to win, M_{win}. In this way, an M_{win}-move puzzle of (N_B, N_R, N_b, N_r) is generated.

We discuss the major advantages and disadvantages of the random placement generation as follows. Regarding the advantages, first, the method is easy to understand and implement. Second, since randomness is involved, the method can generate diverse puzzles. Regarding the disadvantages, first, the probability of getting an endgame puzzle from a random piece placement is not high, which is approximately 20% to 30% in the experiments in Section 5. Second, the method seldom generates challenging puzzles with high M_{win}, e.g., a success rate of 0.02% for 17-move not-revealed puzzles, which will be shown in Section 5. Third, the puzzles may lack aesthetics. More specifically, we consider some aspects that have been studied in chess mating problems to be applicable to Geister endgame puzzles, e.g., containing pieces that are irrelevant to the solutions (i.e., redundant pieces) [5] or missing creative transmutations [27]. We will show a Geister endgame puzzle that contains redundant pieces in Section 7 when discussing future research directions.

4.2. Solver

We employ a solver that builds AND-OR trees [28] to *solve* Geister positions, i.e., to prove whether or not the bottom-side player can win, assuming that the

top-side player's unknown pieces are purple pieces (blue when escaped and red when captured) proposed by Sato [17] and that the top-side player knows the bottom-side player's piece colors. In the trees, the root node represents the given position; OR-nodes represent positions of the bottom-side player's turn and AND-nodes the top-side player's. Each node in the trees has a value of *win*, *loss*, or *unknown* from the bottom-side player's view. In addition, each edge represents a move from the corresponding node.

Fig. 3 depicts part of an AND-OR tree for the position in Fig. 2a. Leaf nodes G, K, and M are the bottom-side player's wins determined by game rules, while leaf node F is a loss. For non-leaf nodes, the initial values are *unknown*, and subtrees are expanded to determine whether the node is a win or a loss under the optimal play. A non-leaf OR-node is a win if at least one of its children is a win, and a loss if all children are losses. Similarly, a non-leaf AND-node is a loss if at least one of its children is a loss, and a win if all children are wins. For example, OR-node I is a win since its child K is a win, and AND-node C is a loss since its child F is a loss. As for AND-node H, all of its children (e.g., node J) can be shown as wins in a similar way to node I, which shows that node H is a win. A node that has been known to be a win is also called *proven*, while a node known to be a loss is called *disproven*.

The solver is based on the depth-first proof-number search (abbr. df-pn) [10], a variant of proof number search (PNS) that aims to reduce memory usage. PNS is a classical algorithm for solving positions, which selects the next nodes to expand according to the *proof numbers* and *disproof numbers* of nodes. Since both PNS and df-pn do not guarantee finding the shortest wins, we further add depth limits to the searches and combine iterative deepening search [29] in order to obtain M_{win}. For a given depth limit, counted from the root node (depth 0), if a node cannot be proven, we regard it as disproven (a loss). For example, with a depth limit of 3, AND-node H in Fig. 3 is judged as disproven since the bottom-side player cannot win within 3 moves from the root.

As for the iterative deepening search, the depth limit starts from three[2] and increases by two[3] until the given position is proven or disproven. In this way, if a position is proven at a depth limit of k, the position is a k-move puzzle since the searches until the depth limit of $(k-2)$ have shown that the bottom-side player cannot win in $(k-2)$ moves. The position in Fig. 2a can be proven by a search with a depth limit of 5, explained as follows. With a depth limit of 5, OR-nodes I and J are proven since leaf nodes K and M are proven. AND-node

[2]Note that 1-move puzzles are intrinsically not generated for two reasons. First, the bottom-side player's blue pieces are never placed on the top-side exits. Second, for capture-win puzzles, the bottom-side player needs to capture at least two pieces since the worst case (purple pieces) assumes the captured pieces to be red when there are unrevealed ones. Also, it is impossible for the bottom-side player to win by two moves. Therefore, it is no problem to skip the depth limits of one and two.

[3]Since it is impossible for the bottom-side player to win by an even number of moves (e.g., 6), there are no k-move Geister endgame puzzles where k is even. Thus, it is no problem to skip the depth limits of even numbers.

H can then be proven since all its children can be proven in similar ways to nodes I and J, and thus OR-node E can be proven. Similarly, all of AND-node B's children can be proven, showing that the bottom-side player can win by 5 moves where the first one is to move the R to e6. In practice, we set a maximum depth limit (or time limit) to prevent the solver from continuing endlessly.

To speed up the solver, we employ two techniques exploiting game-specific knowledge. The first technique judges victory condition 1, i.e., escaping one blue piece, before the player moves. In more detail, the solver checks whether the player to move has one blue piece at one of the player's exit squares. If the condition holds, the player wins without further expanding the nodes. Since it is the player's turn, the player can always escape the blue piece at the exit to win; thus, there is no problem with the correctness. For example, OR-node I in Fig. 3 is the bottom-side player's turn and the B is at the top-side exit of f6. The node can be proven without expanding its children, such as AND-nodes K and L. The judgment has negligible computational costs, especially when bitboards [30] are applied. For not-revealed puzzles, the bottom-side player must win by escaping a blue piece, and we further apply this technique to reduce the depth limits from 3, 5, 7, etc., to 2, 4, 6, etc. For Fig. 3, the root (i.e., the position in Fig. 2a) can be proven by a depth limit of 4 with this technique, and there is no need to expand nodes at depth 5 (i.e., AND-nodes K, L, M, etc.).

The second technique uses heuristics to prune nodes when possible. More specifically, when the bottom-side player is judged unable to win within the remaining moves of the depth limit, the corresponding node is disproven without further search. For not-revealed puzzles, the bottom-side player needs to escape a blue piece from either top-side exit. Thus, a node can be safely judged as disproven upon knowing that the bottom-side player cannot escape any blue piece within the remaining moves. For partly-revealed puzzles, the bottom-side player needs to either escape a blue piece or capture all the top-side player's blue pieces. Thus, a node can be judged as disproven upon knowing that the bottom-side player cannot escape any blue piece *and* cannot capture all the top-side player's blue pieces within the remaining moves. The detailed implementation will be shown in Appendix A.

4.3. Identification of Red-Wall and Capture-Win Puzzles

As mentioned in Subsection 3.3, we define two kinds of special puzzles: red-wall and capture-win. This subsection explains how to find such puzzles to enable further analyses (e.g., the interestingness and difficulty) or applications (e.g., providing to human players). Red-wall puzzles are those utilizing victory condition 3, i.e., having all red pieces captured by the opponent, as discussed in Subsection 3.3.3. For the puzzle in Fig. 2c, after the bottom-side player moves the R at a2 to a1, the top-side player cannot capture the R to escape from a1. Otherwise, the top-side player loses immediately due to capturing all the red pieces of the bottom-side player, which is not the optimal play (longest loss). Assuming victory condition 3 does not exist, the top-side player can capture the R at a1 and escape earlier than the bottom-side player. Namely, the position will no longer be a 7-move puzzle. For a given k-move Geister endgame puzzle,

we propose to identify red-wall puzzles by doing an additional search, where victory condition 3 for the bottom-side player is removed. If the bottom-side player cannot win by k moves in the search, it means that the bottom-side player utilizes victory condition 3 to block the top-side player's moves, i.e., a red-wall puzzle.

Capture-win puzzles have two prerequisites: the puzzle is a partly-revealed one, and at least one of the top-side player's red pieces is revealed, as shown in Fig. 2b. As long as the bottom-side player does not capture all of the revealed red pieces, it is possible to win by capturing all of the top-side player's blue pieces (victory condition 2). For a given k-move partly-revealed puzzle, we propose to identify capture-win puzzles by doing an additional search. In the search, the first technique in Subsection 4.2 (i.e., judging wins by escape before moving) is required, and we set the depth limit to $k-1$. If the bottom-side player cannot win, the puzzle is a capture-win puzzle; otherwise, the bottom-side player should be able to win by escape (victory condition 1). For example, assuming that there is `B` at a4 in Fig. 2b, escaping the `B` from a6 is another solution to the 5-move puzzle. This case is not judged as a capture-win puzzle since it can be proven by the additional search. Namely, the win by escape can be proven by a search with a depth limit of (5-1), and the puzzle is not capture-win.

5. Experiments on Random Placement Generation

Subsection 5.1 presents the experiment settings. Subsection 5.2 demonstrates the effectiveness of the technique of heuristic pruning implemented for the df-pn solver. Subsection 5.3 shows the analyses of random placement generation.

5.1. Experiment Settings

In the experiments, we investigated all possible combinations of piece numbers between 1 to 3. More specifically, piece number combinations of ($N_B \in \{1,2,3\}$, $N_R \in \{1,2,3\}$, $N_b \in \{1,2,3\}$, $N_r \in \{1,2,3\}$) were experimented. Although the maximum number of each piece type is 4, we considered it relatively natural that both players already have some pieces being captured in endgames. For not-revealed puzzles, we studied $3^4 = 81$ combinations. For partly-revealed puzzles, since it is impossible to partly reveal for $N_b = 1$ and $N_r = 1$, we studied $3^4 - 3 \times 3 = 72$ combinations.

As for the df-pn solver, both techniques introduced in Subsection 4.2 were employed unless specified. In addition, a maximum depth limit (or time limit) was needed so that the search would not run infinitely long, as mentioned in Subsection 4.2 (the 4th paragraph). In the experiments, we set the maximum depth limit to 17, which collected 3-move, 5-move, ..., 17-move puzzles. Note that there are intrinsically no 1-move puzzles as discussed in Footnote 2. Also, positions not collected did not necessarily mean that the bottom-side player could not win. It was possible that the positions required higher depth limits

Table 1: Results of solvers with and without the heuristic pruning technique

	Not-revealed		Partly-revealed	
	Time/Trial	#Nodes/Trial	Time/Trial	#Nodes/Trial
No pruning	7.02 s	1,554,550	15.72 s	3,150,892
Pruning	0.37 s	105,926	3.45 s	751,875
Improved rate	94.67%	93.19%	78.07%	76.14%

to be proven. However, as will be shown in Subsection 5.3, the number of puzzles drastically decreased as M_{win} increased. Thus, we expected most of the not-collected positions to be those that the bottom-side player could not win.

5.2. Effectiveness of Heuristic Pruning

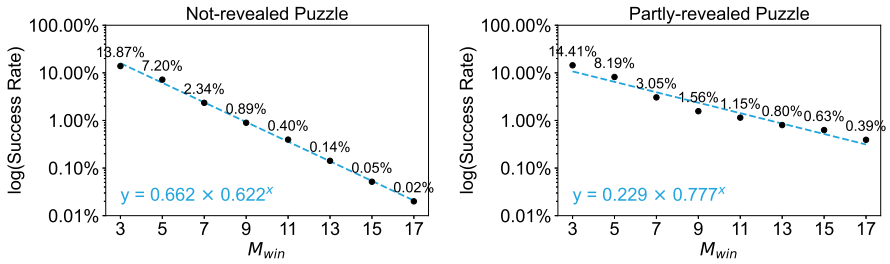
To verify the effectiveness of the heuristic pruning technique implemented in the df-pn solver, we compared the results of solvers with and without the technique. Readers who skipped Subsection 4.2 can skip this subsection. For each combination of (N_B, N_R, N_b, N_r), we performed 50 trials, where one trial means that the given number of pieces are randomly placed on the board to create a position. We ran the experiments on a machine equipped with Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz and 32.0 GB memory. The solver used a single thread without parallelization.

We made the comparisons based on two metrics: the average execution time per trial and the average number of expanded nodes per trial. Table 1 shows the results of 50 trials to generate not-revealed and partly-revealed puzzles. The improved rates in the tables are calculated by (("No pruning" - "Pruning") / "No pruning"). The results confirmed that the heuristic pruning technique improved the search efficiency a lot, especially for not-revealed cases. We considered it reasonable that the partly-revealed cases had lower improved rates since the criteria were more strict (cannot escape *and* cannot capture all the top-side player's blue pieces). When further looking into different piece number combinations (N_B, N_R, N_b, N_r), i.e., (1, 1, 1, 1), (1, 1, 1, 2), ..., and (3, 3, 3, 3), we confirmed that the efficiency improvement was generally effective across different combinations.

In addition, the results showed that each trial to generate not-revealed puzzles was less costly than partly-revealed puzzles. One possible reason was the first technique (i.e., judging escapes before moving). As discussed in Subsection 4.2 (the 5th paragraph), the technique was used to reduce the depth limits to 2, 4, 6, etc., in the searches for generating not-revealed puzzles, where the depth limits for partly-revealed ones were 3, 5, 7, etc.

5.3. Analyses of Puzzle Generation

In order to know which kinds of puzzles are likely generated by the random generation in terms of M_{win} and the piece numbers, we made analyses based on success rates of generating puzzles from trials. For example, if 10 puzzles are generated from 50 trials, the success rate is 10/50=20%. To get



(a) (b)

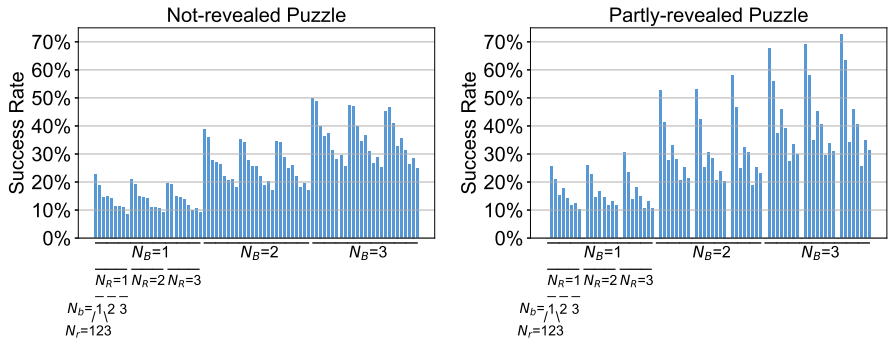
Figure 4: Success rates according to M_{win} for (a) not-revealed and (b) partly-revealed puzzles.

statistically significant results of success rates, we increased the trial numbers for each piece number combination (50 in the previous experiment) in this experiment. In more detail, for generating not-revealed puzzles, we performed 10,000 trials for each piece number combination; for partly-revealed puzzles, we performed only 2,500 trials since it cost much more time. The overall success rate, along with the 95% confidence interval, of not-revealed puzzles was 24.92%±0.09% (=201,829/810,000) and that of partly-revealed puzzles was 30.18%±0.21% (=54,330/180,000).

Fig. 4 plots the success rates in logarithmic scales for each $M_{win} \in \{3, 5, ..., 17\}$. The exponential regression curves and functions are also shown in the figures. As expected, when M_{win} increased, the success rates dropped drastically. Moreover, we observed that success rates had approximately exponential relations to M_{win} in both kinds of not-revealed and partly-revealed puzzles. Overall, it was easier to generate partly-revealed puzzles than not-revealed puzzles from random trials. Especially, the difference in success rates became bigger as M_{win} increased (about 20 times when M_{win} was 17). Since all of the top-side player's pieces in not-revealed puzzles are treated as purple pieces (the worst case), there are more restrictions for positions to become endgame puzzles than partly-revealed puzzles. For example, when generating not-revealed puzzles, if one of the top-side player's pieces is put on a bottom-side exit without the bottom-side player's pieces guarding that exit, the position never becomes an endgame puzzle. In contrast, for partly-revealed puzzles, if the top-side player's piece is a revealed red piece, even when the exit is not guarded, the bottom-side player will not lose immediately. Therefore, it is reasonable that partly-revealed puzzles had higher success rates.

Fig. 5 shows the success rates of all the experimented piece number combinations, ($N_B \in \{1, 2, 3\}$, $N_R \in \{1, 2, 3\}$, $N_b \in \{1, 2, 3\}$, $N_r \in \{1, 2, 3\}$), for not-revealed and partly-revealed puzzles. In addition, we performed multiple linear regression to analyze the relations between piece numbers and success rates. The success rates of not-revealed puzzles (S^N) and partly-revealed puzzles (S^P) could be approximated by

$$S^N \approx 0.222 + 0.107 \times N_B - 0.006 \times N_R - 0.067 \times N_b - 0.021 \times N_r, \quad (1)$$



(a) (b)

Figure 5: Success rates of piece number combinations of (N_B, N_R, N_b, N_r) for (a) not-revealed and (b) partly-revealed puzzles.

$$S^P \approx 0.266 + 0.132 \times N_B + 0.006 \times N_R - 0.118 \times N_b + 0.005 \times N_r, \qquad (2)$$

respectively. The coefficients of determination (R^2) for the two equations were 0.948 and 0.869, meaning that the equations could make good predictions. For both kinds of puzzles, the success rates increased as N_B increased while decreased as N_b increased. Namely, when the bottom-side player had more blue pieces or when the top-side player had fewer blue pieces, it was easier to generate puzzles. We considered the results reasonable since the one owning more blue pieces is more likely to win the game.

5.3.1. Analyses of Red-Wall and Capture-Win Puzzles

In Subsection 3.3, we proposed two kinds of special puzzles, i.e., red-wall and capture-win, and in Subsection 4.3, we proposed methods to find such puzzles. We used the proposed methods to check each generated puzzle to see whether the puzzle satisfied special criteria. Fig. 6 shows the results of red-wall and capture-win puzzles, including the overall ratios of such special puzzles among generated puzzles (shown in figure titles) and each $M_{win} \in \{3, 5, ..., 17\}$'s numbers (gray bars) and ratios (curves) of special puzzles. For ratios, the 95% confidence intervals were also presented. For example, among 706 partly-revealed 17-move puzzles, 184 were red-wall, and the ratio was 184/706=26.06%±3.24%.

For partly-revealed puzzles, most M_{win} had approximately 150 red-wall or capture-win puzzles (Figs. 6b and 6c), and the ratios of such special puzzles increased as M_{win} increased. For not-revealed puzzles, the ratios of red-wall puzzles also increased as M_{win} increased, though the numbers of puzzles drastically dropped. One possible reason was that not-revealed puzzles with high M_{win} were infrequently generated (Fig. 4a). Generally, it was not too difficult to generate special puzzles with middle or relatively high M_{win} (say 9 to 17), especially partly-revealed ones. Since puzzles with low M_{win} (say $\leq$7) are usually too trivial even for beginners and are not desired, we considered these results promising for use cases that provide human players with special puzzles.

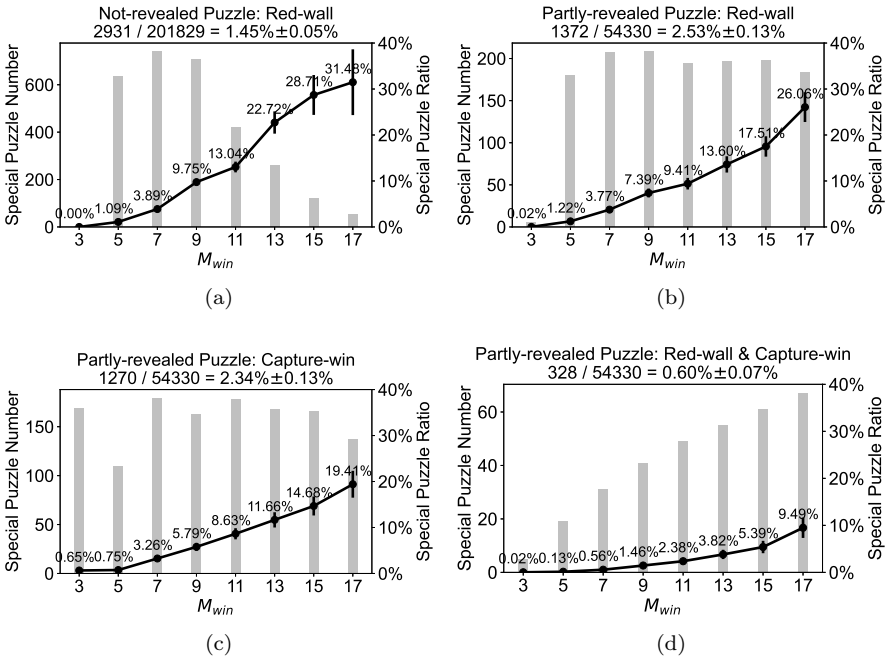


Figure 6: Special puzzles' numbers and the ratios among generated puzzles for each M_{win} for (a) not-revealed red-wall puzzles, (b) partly-revealed red-wall puzzles, (c) partly-revealed capture-win puzzles, and (d) partly-revealed puzzles of both red-wall and capture-win.

6. Examples of Geister Endgame Puzzles

In this section, we show some examples of Geister endgame puzzles. We also show an example solution for each puzzle, where a move is denoted by [piece][coordinate][direction]. These puzzles were from a preliminary experiment where we asked beginners to evaluate the interestingness and difficulty of Geister endgame puzzles. In this section, all puzzles except one for comparison purposes were evaluated to be interesting in the preliminary experiment. More details of the preliminary experiments can be found in Appendix B.

Figs. 7 to 12 show puzzles and example solutions of Geister endgame puzzles with different categories, which is summarized in Table 2. More specifically, Fig. 7 shows three examples of not-revealed puzzles that are not red-wall. Fig. 8 shows three examples of not-revealed red-wall puzzles. Fig. 9 shows three examples of partly-revealed puzzles that are neither red-wall nor capture-win. Fig. 10 shows three examples of partly-revealed puzzles that are red-wall but not capture-win. Fig. 11 shows three examples of partly-revealed puzzles that are capture-win but not red-wall. Fig. 12 shows three examples of partly-revealed puzzles that are both red-wall and capture-win.

Fig. 7b was evaluated to be the most interesting in the preliminary experiments. The main story is the offense and defense on the top-left corner. The

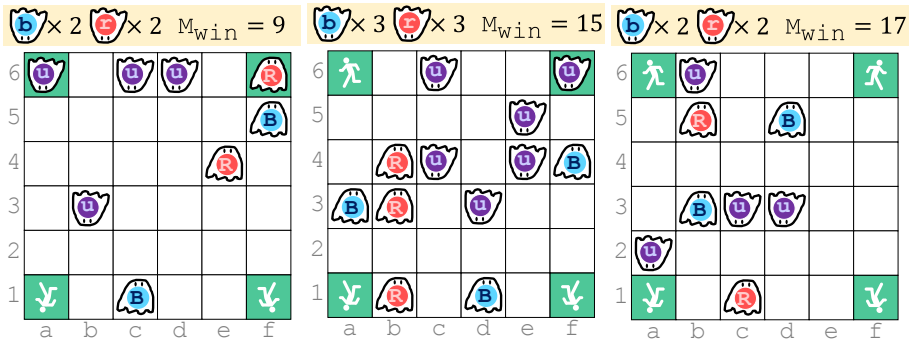


Figure 7: Examples of not-revealed Geister endgame puzzles that are not red-wall.

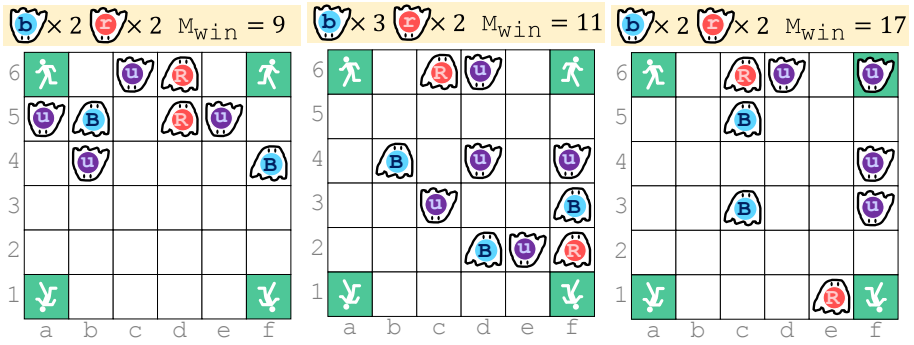


Figure 8: Examples of not-revealed red-wall Geister endgame puzzles.

Example Puzzles	Revealed	Red-Wall	Capture-Win
Fig. 7	Not	No	–
Fig. 8	Not	Yes	–
Fig. 9	Partly	No	No
Fig. 10	Partly	Yes	No
Fig. 11	Partly	No	Yes
Fig. 12	Partly	Yes	Yes

Table 2: A summary of example puzzles in Figs. 7 to 12

bottom-side player needs to utilize the two `R`'s at b4 and b3 for supporting the `B` at a3 to escape. At first glance, the bottom-side player can take the `u` at c4; however, it is not the first priority. Instead, the bottom-side player should move `R` to b5 for curbing the `u` at c6. From the human players' feedback, they felt surprised by such a move that did not capture the opponent's piece immediately when they could.

Fig. 7c was evaluated to be the most difficult, though not interesting, in the preliminary experiments. At first glance, players may think it is an easy problem to escape from f6. In fact, the bottom-side player needs to protect the exit at a1 at first, attract the `u` at b6 by the `B` at d5, and then escape the `B` at b3 from a6 with the support of the `R` at b5. Among the 17 moves of the bottom-side and top-side players, several playing skills are involved, and the way to escape is not straightforward.

Overall, we can see that there are diverse Geister endgame puzzles. Some involve complex piece exchanges, some involve unstraightforward escapes, some need to utilize the bottom-side player's red pieces (red-wall), etc. In addition, we confirmed through preliminary experiments that we could find interesting Geister puzzles. We believe that such puzzles can entertain human players and help them improve their playing skills.

7. Conclusions and Future Work

In this paper, we proposed endgame puzzles for the imperfect information game Geister, aiming to assist human players in improving their playing skills or serve as entertainment. We defined two kinds of puzzles according to whether the opponent's piece colors are revealed or not. Assuming the bottom-side player's view, not-revealed puzzles follow Geister's game rules that the top-side player's piece colors are not revealed. Partly-revealed puzzles reflect the fact that expert players can usually infer some of the opponent's piece colors in endgames. The goal of the puzzles is to find the shortest winning moves under two assumptions of worst cases: (i) the top-side player's unrevealed pieces are regarded as purple pieces (becoming red when captured and blue when escaped), and (ii) the top-side player knows the bottom-side player's piece colors. We consider that the concept of assuming the worst cases can be applied to create puzzles for other imperfect information games.

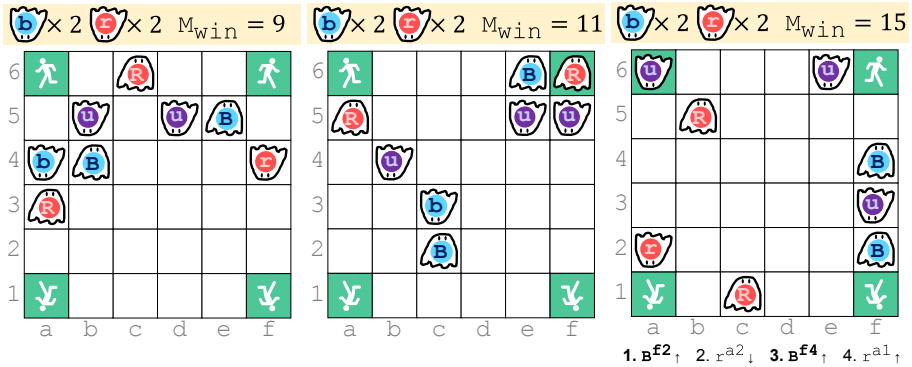


Figure 9: Examples of partly-revealed Geister endgame puzzles that are neither red-wall nor capture-win.

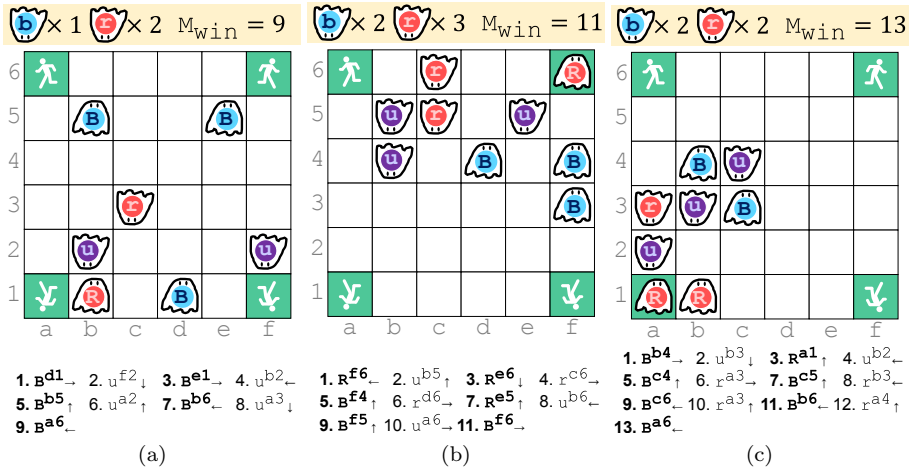


Figure 10: Examples of partly-revealed Geister endgame puzzles that are red-wall but not capture-win.

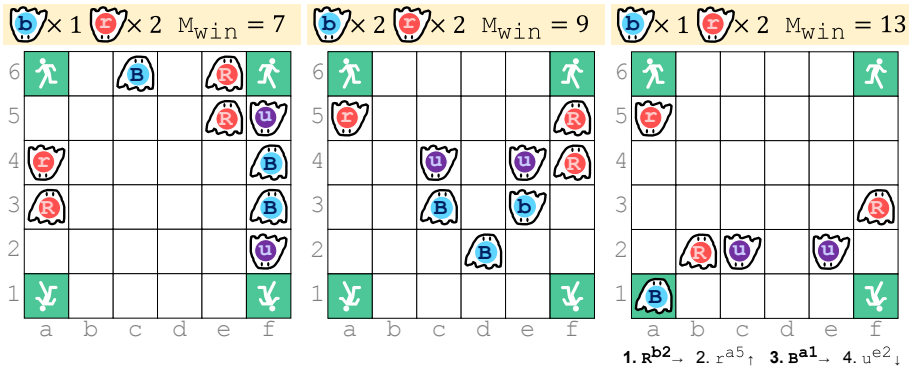


Figure 11: Examples of partly-revealed Geister endgame puzzles that are capture-win but not red-wall.

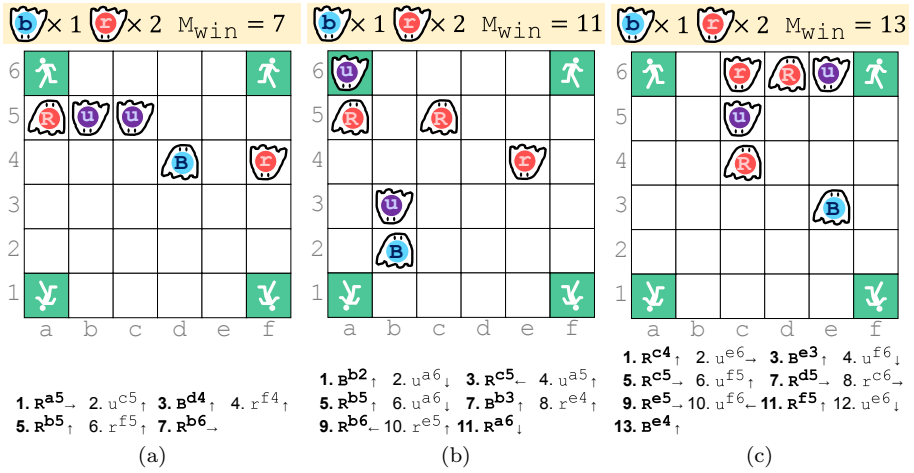


Figure 12: Examples of partly-revealed Geister endgame puzzles that are both red-wall and capture-win.

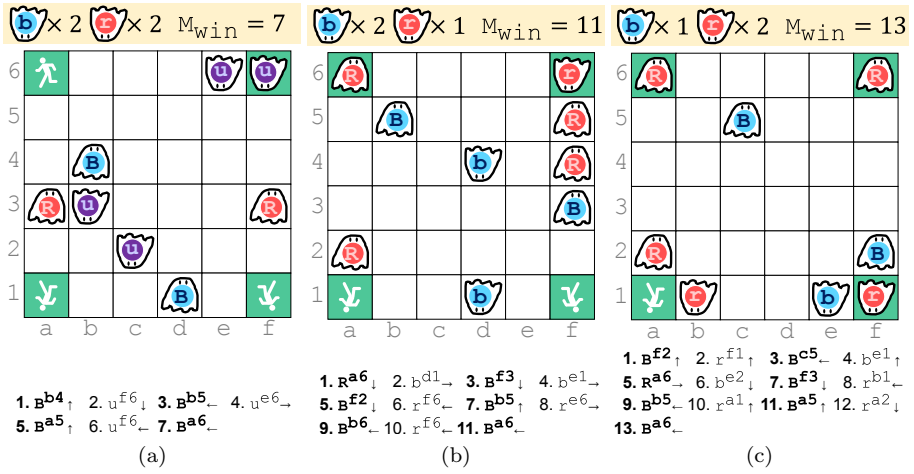


Figure 13: (a) A puzzle with redundant pieces and puzzles (b) with and (c) without the opponent's futile offense.

Employing a solver combining the depth-first proof-number search and the iterative deepening search, we generated Geister endgame puzzles using the random placement method. In more detail, we randomly placed pieces of given numbers on the board and used the solver to check whether the bottom-side player could win in a certain number of moves. The experiments showed that the success rates of generation decreased approximately exponentially as the number of moves to win increased. In addition, the results suggested that the success rates were higher when the bottom-side player had more blue pieces or when the top-side player had fewer blue pieces, which was understandable. In the experiments, we also identified special puzzles, i.e., red-wall and blue-capture. Although the overall ratio was low (approximately 1% to 2%), we confirmed that such puzzles could be generated even using simple methods like random placement.

For future research, one direction is to propose more types of Geister endgame puzzles that require specific playing skills, such as sacrificing the player's own pieces. Another direction is to investigate the aesthetics of Geister endgame puzzles. The aesthetics of chess problems [31, 5] or tsume-shogi [21, 32] is widely studied. For example, in chess problems or tsume-shogi, it is preferred that the first move in the solution is unique, which is not considered in our definition of Geister puzzles in this paper. In addition, redundant pieces are usually undesirable. For the Geister puzzle example of Fig. 13a, the solution is to escape the B at b4 from a6. Even if some of the other pieces are removed, the bottom-side player still wins in the same way. How to define redundant pieces in Geister endgame puzzles is worth investigating.

Another topic is futile offense, which is also undesirable in general. Figs. 13b and 13c are example puzzles that we manually made to show futile offense. For

simplicity of consideration, all the top-side player's pieces are revealed. For the puzzle in Fig. 13b, the bottom-side player alone needs 4 moves to escape the B at b5 from a6; thus, it is intrinsically a 7-move puzzle. However, it is actually an 11-move puzzle under the current definition since the top-side player tries to escape the b at d1 from f1 and the bottom-side player alone must spend 2 moves (a total of 4 moves, including the top-side player's) to prevent the escape. Such offense merely increases the total number of moves to win and may be annoying to players. In contrast, we consider the puzzle in Fig. 13c a good instance. For the bottom-right corner, if the bottom-side player does nothing, the top-side player alone costs 3 moves to win ($\mathtt{r}^{\mathrm{f1}}\uparrow$, $\mathtt{b}^{\mathrm{e1}}\rightarrow$, $\mathtt{b}^{\mathrm{f1}}\rightarrow$); thus, the bottom-side player needs to move the B at f2 to f3. At this time point, the top-side player alone takes 6 moves ($\mathtt{b}^{\mathrm{e1}}\uparrow$, $\mathtt{r}^{\mathrm{f1}}\uparrow$, $\mathtt{b}^{\mathrm{e2}}\downarrow$, $\mathtt{b}^{\mathrm{e1}}\rightarrow$, $\mathtt{b}^{\mathrm{f1}}\rightarrow$) to win. Meanwhile, the bottom-side player alone takes 5 moves to escape the B at c5 from a6, including moving the R at a6 to a5. Namely, the bottom-side player is only one move ahead of the top-side player to win. We consider such an offense to be non-futile, and how to define it algorithmically is worth investigating.

In addition to the definitions and aesthetics of puzzles, another future research direction is to employ new methods for puzzle generation that can more efficiently generate puzzles with high numbers of moves to win. For example, it is promising to employ the reverse method [21], which starts from a winning position and moves pieces in a backward way. It may be interesting to try genetic algorithms [33] or machine learning methods [34]. Regarding evaluations, it is valuable to generate interesting puzzles with proper difficulty for training human players and analyze how their playing skills are improved.

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.

Funding

This work was supported by JSPS KAKENHI Grant Numbers JP18H03347, JP20K12121, JP20K19946, and JP22K12339.

Appendix A. Implementations of Pruning Heuristics

This appendix explains the pruning heuristics for not-revealed puzzles and partly-revealed puzzles. The heuristics for the two categories are slightly different and are separately explained. Note that the first technique in Subsection 4.2 (i.e., judging wins by escape before actually moving) can be used together with these heuristics, but in the following, we assume cases without the first technique for simplicity of discussions.

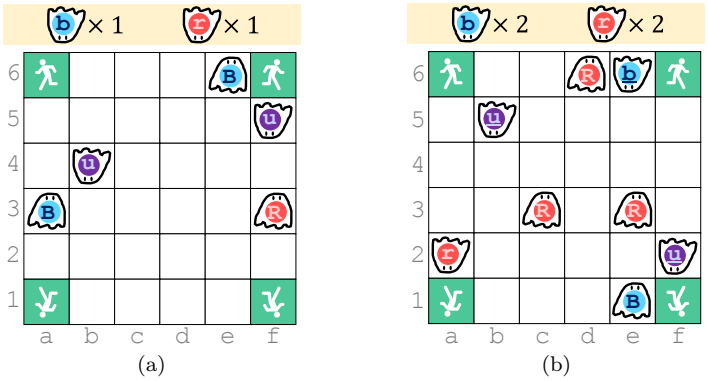


Figure A.14: Examples of pruning heuristics for (a) escape and (b) capture.

The notation that will be used in this appendix is summarized as follows. Let $\{B\}$ denote the set of bottom-side player's blue pieces, $\{R\}$ the red pieces, $\{b\}$ the top-side player's revealed blue pieces, $\{r\}$ the revealed red pieces, $\{u\}$ the not-revealed pieces, and $d(x, y)$ the Manhattan distance between x and y.

For not-revealed puzzles, a node can be safely judged as disproven upon knowing that the bottom-side player cannot escape any blue piece within the remaining moves. More specifically, we made the judgment based on the shortest Manhattan distance between the bottom-side player's blue pieces and the top-side exits, i.e., $\min_{p\in\{B\},e\in E^B} d(p, e)$, where E^B is the set of the top-side exits a6 and f6. For the example of Fig. A.14a, the shortest distance is 1 between the `B` at e6 and the exit at f6. Even if the bottom-side player tries to escape the `B` at e6 from f6 and the top-side player does not capture it, the escape requires 3 moves, including both players'. According to whether it is the bottom-side player to move or not, we regard the corresponding node as disproven when the remaining number of moves to the depth limit is less than 3 or 3+1.

For partly-revealed puzzles, a node can be judged as disproven upon knowing that the bottom-side player cannot escape any blue piece *and* cannot capture all the top-side player's blue pieces within the remaining moves. The calculation of escape is the same as not-revealed puzzles. As for the latter condition (capture), it is always true when the top-side player does not have any revealed red piece. The following assumes that the top-side player has at least one revealed red piece. In order to win by capture, the bottom-side player needs to capture all the top-side player's blue pieces and not-revealed pieces (i.e., $\{b\}\cup\{u\}$). For the example in Fig. A.14b, the bottom-side player should capture the `b` at e6 and the two `u`'s at b5 and f2 (i.e., those with underscores). We propose to calculate the least required moves to win by taking the maximum of the following two cases.

- $(\min_{p\in\{B\}\cup\{R\},q\in\{b\}\cup\{u\}} d(p, q)) + 2 \times (|\{b\} \cup \{u\}| - 1)$, where $|S|$ is the size of set S. The Manhattan distance between p and q is the number of

moves that the two pieces move toward each other and meet at the same square, regardless of who is going to move. The minimum distance is the least number of moves to capture a `b` or a `u`. For each of the other `b`'s and `u`'s, at least two moves are required for the capture, one move for the top-side player and the other for the bottom-side player to capture. For the example of Fig. A.14b, the least number of required moves is $1+2\times 2=5$, where 1 is for the `b` at e6.

- $\max_{q\in\{b\}\cup\{u\}}(\min_{p\in\{B\}\cup\{R\}} d(p,q))$. It is the least number of moves to capture the farthest `b` or `u`. For the example of Fig. A.14b, the least number of required moves is $max(1,2,3)=3$, where 3 is for the `u` at b5.

Although it is possible to design more sophisticated rules to prune more nodes, dedication to pruning heuristics is not the main purpose of this work. Also, the experiments in Subsection 5.2 have shown that the presented heuristics improved efficiency effectively. Thus, the heuristics are kept simple. An example of an advanced pruning heuristic is that if the top-side exits are guarded by the top-side player, as in Fig. A.14a, the bottom-side player should spend more moves to escape.

Appendix B. Interestingness and Difficulty of Puzzles

This appendix presents an investigation of Geister endgame puzzles' interestingness and difficulty through a human subject experiment. We expect that generated puzzles vary a lot; however, for future applications of introducing Geister endgame puzzles to human players for entertainment or practicing playing skills, interesting puzzles with proper difficulty for the target players are desirable. In this work, we targeted beginners, and in order to know the interestingness and difficulty of puzzles from beginners' views, we conducted a subject experiment described in Appendix B.1. Moreover, we performed supervised learning to learn the interestingness and difficulty, explained in Appendix B.2.

Appendix B.1. Human Players' Evaluations

We employed eight Geister beginners to evaluate 100 endgame puzzles. We selected the 100 puzzles subjectively in a way that we expected to include puzzles with diverse interestingness and difficulty[4]. In addition, most of the puzzles

[4]Our subjective selection may have introduced biases into the puzzle set. Random selection would allow us to do objective analyses; however, doing experiments using randomly selected puzzles was practically challenging. When selecting purely randomly from generated puzzles, high M_{win}-move puzzles were few, as shown in Fig. 4, and were hardly selected. Similarly, the overall ratios of red-wall and capture-win puzzles were low and hardly selected. To collect sufficiently many evaluations of high M_{win}-move puzzles or red-wall/capture-win puzzles, we would have to provide the participants with a large number of puzzles, which was infeasible. Or we might randomly select high M_{win}-move puzzles or red-wall/capture-win puzzles from generated ones; however, this raised another problem: we would need to decide how many puzzles we wanted to collect for each category, which also introduces biases. Considering the experiment cost, we selected the 100 puzzles instead of using a random selection while we tried our best to keep the puzzles as diverse as possible.

Table B.3: The number of puzzles, average interestingness, and average difficulty according to M_{win}

M_{win}	5	7	9	11	13	15	17	19
#Puzzles	2	10	19	18	29	17	3	2
Avg. Inter.	-1.38	-0.48	-0.13	-0.14	+0.12	-0.11	+0.21	+0.06
Avg. Diffi.	-1.88	-1.44	-0.88	-0.40	+0.05	+0.32	+1.42	+1.00

had moderate M_{win} to prevent players from getting too bored or too frustrated soon during the experiment. Table B.3 lists the numbers of puzzles for each $M_{win} \in \{5, 7, ..., 19\}$, and Table B.4 lists the number of puzzles according to categories (e.g., not-revealed vs. partly-revealed).

For each player, the order of the 100 puzzles was different. For each puzzle, a player had at most 90 seconds to solve the puzzle. After then, no matter whether the player solved the puzzle or not, an example solution was displayed. We asked the players to evaluate the puzzles' interestingness and difficulty separately using a five-level Likert scale, {-2, -1, 0, +1, +2}, where -2 meant boring/easy and +2 meant interesting/difficult. For each puzzle, we averaged the evaluation values from the eight players.

Table B.3 lists the interestingness and difficulty for each M_{win} averaged from the corresponding M_{win}-move puzzles. As expected, puzzles with higher M_{win} tended to be more difficult. In contrast, the increasing trend of interestingness was unclear. More specifically, puzzles with low M_{win} tended to be boring, but those with high M_{win} were not necessarily interesting. A similar phenomenon has also been reported by Margulies[5] in a study of chess mating problems, which showed that difficult moves usually required more calculations but were not necessarily beautiful moves from experts' views.

Table B.4 lists the average M_{win}, interestingness, and difficulty according to categories (not-revealed or partly-revealed, red-wall or not, and capture-win or not). We observed that different categories had similar average M_{win}, implying that the tendencies were less likely to be caused by M_{win}. The average interestingness showed that partly-revealed, red-wall, or capture-win puzzles were more interesting than those not, where the difference between red-wall and not-red-wall was statistically significant (a p-value of 0.01 for Welch's t-test). The average difficulty showed no significant differences among different categories.

Fig. 7b shows the most interesting puzzle evaluated by the human players in the subject experiment, which had an interestingness of 1.25. We have presented the main story of the puzzle in Section 6 (3rd paragraph). Fig. 7c shows the most difficult puzzle evaluated by the human players in the subject experiment, which had a difficulty of 1.88. We have discussed the solution of the puzzle in Section 6 (4th paragraph). The puzzle received an interestingness of -0.38,

[5]S. Margulies, Principles of beauty, Psychological Reports 41 (1) (1977) 3–11. 10.2466/pr0.1977.41.1.3

Table B.4: The number of puzzles, average M_{win}, average interestingness, and average difficulty according to categories

	Revealed		Red-wall		Capture-win (in partly-revealed)	
	Not	Partly	Yes	No	Yes	No
#Puzzles	62	38	36	64	19	19
Avg. M_{win}	12.10	11.05	12.17	11.44	10.89	11.21
Avg. Inter.	-0.19	+0.04	+0.11	-0.22	+0.11	-0.03
Avg. Diffi.	-0.33	-0.21	-0.17	-0.35	-0.24	-0.18

providing an example that difficult puzzles were not necessarily interesting. We suspected that if intermediate or higher-level players played this puzzle, or if the beginners were given a longer time, the puzzle might be evaluated as more interesting.

Appendix B.2. Supervised Learning on Interestingness and Difficulty

Based on the evaluations from human players, we applied supervised learning to learn the interestingness and difficulty of Geister endgame puzzles. If the models learned well, it would be possible to automatically generate interesting puzzles or puzzles with proper difficulty for the target players. We proposed 32 features that might relate to the two metrics of puzzles and then further reduced the number to 14 according to some preliminary experiments. Appendix B.2.1 will explain more about the features, such as the total number of nodes expanded by the solver.

As for the learning algorithm, we employed the LightGBM framework (version 2.2.3)[6]. It is a gradient boosting framework based on decision trees. We trained models for predicting interestingness and difficulty separately. The objective was to minimize the mean squared errors (MSE) between the predictions and human players' average evaluations of puzzles. We conducted leave-one-out cross-validation to select hyper-parameters of the algorithm. The learning rate (learning_rate) was set to 0.01 and the number of boosting iterations (num_iterations) to 1000. The rest followed the default settings. After learning, the models were evaluated by two metrics commonly used in regression problems: the root-mean-square error (abbr. RMSE, the lower the better) and the coefficient of determination (abbr. R^2, the higher the better). Appendix B.2.2 will show the results.

Appendix B.2.1. Features of Geister Endgame Puzzles

This subsection lists the 32 proposed features for Geister endgame puzzles that might relate to puzzles' interestingness and difficulty. The 14 features shown in italics were used to get the results in Appendix B.2, selected by

[6] `https://lightgbm.readthedocs.io/en/latest`. (Accessed: Apr. 2rd, 2021)

some preliminary experiments. Let N_B, N_R, N_b, and N_r denote the numbers of the bottom-side player's blue pieces, the bottom-side player's red pieces, the top-side player's blue pieces, and the top-side player's red pieces, respectively. 8 features related to piece numbers are (1) N_B, (2) N_R, (3) N_b, (4) N_r, (5) $N_B + N_R$, (6) $N_b + N_r$, (7) the number of the top-side player's revealed blue pieces, and *(8) the number of the top-side player's revealed red pieces.*

Feature *(9)* is *the least number of moves for the bottom-side player to win under optimal play (i.e., M_{win}).*

Let $\{B\}$ denote the set of bottom-side player's blue pieces, $\{R\}$ the red pieces, $\{b\}$ the top-side player's revealed blue pieces, $\{r\}$ the revealed red pieces, $\{u\}$ the not-revealed pieces, E^B the exits for the bottom-side player's blue pieces `B` (i.e., top-side exits a6 and f6), E^b the exits for the top-side player's blue pieces `b` (i.e., bottom-side exits a1 and f1), and $d(x, y)$ the Manhattan distance between x and y. 4 features related to Manhattan distances between pieces and exits are (10) the shortest Manhattan distance between the bottom-side player's blue pieces and their exits, i.e., $\min_{p\in\{B\},e\in E^B} d(p, e)$, also used in the pruning heuristics for escapes, (11) the shortest Manhattan distance between the top-side player's blue and not-revealed pieces and their exits, i.e., $\min_{p\in\{b\}\cup\{u\},e\in E^b} d(p, e)$, *(12) the shortest Manhattan distance between the top-side player's pieces and the top-side exits, i.e.,* $\min_{p\in\{b\}\cup\{u\}\cup\{r\},e\in E^B} d(p, e)$, and (13) the shortest Manhattan distance between the bottom-side player's pieces and the bottom-side exits, i.e., $\min_{p\in\{B\}\cup\{R\},e\in E^b} d(p, e)$. 2 features related to Manhattan distances between pieces, also used in the pruning heuristics for captures, are (14) $\min_{q\in\{b\}\cup\{u\}}(\min_{p\in\{B\}\cup\{R\}} d(p, q))$, and (15) $\max_{q\in\{b\}\cup\{u\}}(\min_{p\in\{B\}\cup\{R\}} d(p, q))$.

For features related to puzzles' solutions, the solutions mean the first ones found by the solver under both players' optimal plays (i.e., the bottom-side player's shortest wins and the top-side player's longest losses). Note that the solutions may not be unique and the found ones may vary according to different implementations of the solver. For puzzles where more than one solution exists, which solution is the most appropriate one remains an open question. Feature *(16)* is *the number of moves in the solution that is not for the "main purpose".* For not-capture-win puzzles, the main purpose is to escape a blue piece, and this feature counts the moves that the bottom-side player does not move the escaped blue piece toward the target exit[7]. For capture-win puzzles, the main purpose is to capture all the top-side player's blue and not-revealed pieces. This feature counts the moves that the bottom-side player neither captures the top-side player's blue or not-revealed pieces nor moves toward the next captured piece.

2 features related to proof numbers and disproof numbers are *(17) the root node's maximum proof number when the puzzle is proven* and *(18) the root node's maximum disproof number right before the puzzle is proven.* These features were

[7]It may be better to separate the moves into those preventing the top-side player's escape and those supporting the bottom-side player's escape, though not easy to define.

also used by Ishitobi [32] for evaluating tsume-shogi puzzles. In the normal df-pn, the root node's proof and disproof numbers are recalculated only when the updates reach the root node. Following Ishitobi [32]'s method, we recalculate the root node's proof and disproof numbers upon the expansion of leaf nodes to obtain intermediate values. Note that once the root node is proven, the disproof number becomes ∞, and we use the maximum value before becoming ∞. A higher maximum proof number means more variations to consider for the bottom-side player's win. A higher maximum disproof number means more variations to consider for the top-side player to avoid losing.

2 features related to node numbers are *(19) the total number of nodes expanded by the solver, including all depth limits until the puzzle is proven*, and *(20) the number of expanded nodes of the depth limit that the puzzle is proven.*

3 features related to captured piece numbers are (21) the number of the bottom-side player's pieces captured in the solution, *(22) the number of the top-side player's pieces captured in the solution*, and *(23) the number of captured pieces in the solution.*

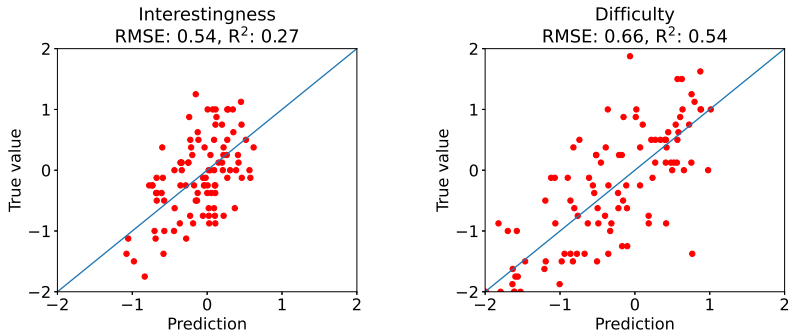
2 features are related to moved piece numbers. Feature *(24)* is *the number of different pieces of the bottom-side player that are moved in the solution.* A higher number means that the bottom-side player has fewer redundant pieces and needs to take more pieces into consideration. Counted in a similar way, feature (25) is the number for the top-side player.

4 features are related to piece arrangements on the board. Feature (26) is the number of the bottom-side player's pieces in the top half of the board, where the number is further divided by $(N_B + N_R)$. If the value is high, the puzzles may give players an impression of attacking the top-side player's area. Similarly, feature *(27)* is *the number of the top-side player's pieces in the bottom half of the board, where the number is further divided by* $(N_b + N_r)$. If the value is high, the puzzles may give players an impression of being attacked by the top-side player. Feature (28) is the number of pieces in the top half, where the number is further divided by the total number of pieces. A value close to 1 (or 0) means that all pieces are in the top (or bottom) half, which may make players feel unnatural or interesting. Similarly, feature (29) counts the left half.

Finally, 3 features related to puzzle categories are *(30) whether the puzzle is not-revealed or partly-revealed*, *(31) whether the puzzle is red-wall or not*, and (32) whether the puzzle is capture-win or not.

Appendix B.2.2. Prediction Results of Interestingness and Difficulty

Fig. B.15 shows the prediction results of the models in the leave-one-out cross-validation. For predicting interestingness, the RMSE was 0.54 and the R^2 was 0.27; for predicting difficulty, the RMSE was 0.66 and the R^2 was 0.54. In both cases, the models could roughly reflect human players' evaluations. Note that it is meaningless to directly compare the RMSE of the two models since the data of interestingness and difficulty had different distributions. Although the difficulty model had a higher RMSE, the R^2 showed that it could make better predictions on the data than the interestingness model. Compared to difficulty, interestingness was a more subjective evaluation, which we expected



(a) (b)

Figure B.15: Leave-one-out cross-validation results of Geister endgame puzzles' (a) interestingness and (b) difficulty.

to be harder to predict. Nevertheless, we considered that both models could be improved by increasing the number of human subjects or the number of puzzles or including advanced features.

Although the models still had room to improve, they could be used to select puzzles in the following senses. The puzzles were likely interesting when the predicted interestingness values were higher than some threshold (say 0.25). As for difficulty, puzzles predicted to be easy (say $\leq$-1) were indeed relatively easy, and those predicted to be difficult (say $\geq$0.5) were likely difficult.

We also analyzed the importance of each feature, which was calculated in the LightGBM framework based on the total gains of splits that used the feature. For interestingness prediction, the most important feature was the number of captured pieces in the first solution found by the solver. We suspected that captures (no matter whose) are attractive to beginners. For difficulty prediction, the most important feature was the total number of nodes expanded by the solver during the search. Although human players usually do not consider as many nodes as computers, we suspected that human players need to consider more situations for puzzles that require more nodes to be proven.

References

[1] D. Silver, T. Hubert, et al., A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play, Science 352 (6419) (2018) 1140–1144. doi:10.1126/science.aar6404.

[2] N. Shaker, J. Togelius, M. J. Nelson, Procedural Content Generation in Games, Springer International Publishing, 2016. doi:10.1007/978-3-319-42716-4.

[3] M. Schlosser, Computers and chess-problem composition, ICGA Journal 11 (4) (1988) 151–155. doi:10.3233/ICG-1988-11404.

[4] Y. HaCohen-Kerner, F. Fainshtein, A deep improver of two-move chess mate problems, Cybernetics and Systems 37 (5) (2006) 443–462. doi:10.1080/01969720600683387.

[5] A. Iqbal, A computational method of optimizing chess compositions to enhance aesthetic appeal, in: 2021 2nd International Conference on Artificial Intelligence and Data Sciences (AiDAS), 2021. doi:10.1109/aidas53897.2021.9574145.

[6] M. Bizjak, M. Guid, Automatic recognition of similar chess motifs, in: The 17th International Conference on Advances in Computer Games (ACG 2021), 2021, pp. 131–141. doi:10.1007/978-3-031-11488-5_12.

[7] T. Wolf, The program GoTools and its computer-generated tsume Go database, in: The Proceedings of the First Game Programming Workshop in Japan, 1994, pp. 84–96.

[8] BoardGameGeek, Phantoms vs Phantoms — Board Game — BoardGameGeek, accessed: Dec. 17, 2023.
URL https://boardgamegeek.com/boardgame/2290/phantoms-vs-phantoms

[9] BoardGameGeek, Stratego — board game — boardgamegeek, accessed: Dec. 17, 2023.
URL https://boardgamegeek.com/boardgame/1917/stratego

[10] A. Nagai, Df-pn algorithm for searching AND/OR trees and its applications, Ph.D. thesis, University of Tokyo, Tokyo, Japan (2002).

[11] N. Kawakami, T. Hashimoto, Research of Geister ai using search for complete information games, in: Proceedings of the 23rd Game Programming Workshop 2018, 2018, pp. 35–42.

[12] F. Aiolli, C. E. Palazzi, Enhancing artificial intelligence in games by learning the opponent's playing style, in: New Frontiers for Entertainment Computing, 2008, pp. 1–10. doi:10.1007/978-0-387-09701-5_1.

[13] S. S. Farooq, H. Park, K.-J. Kim, Inference of opponent's uncertain states in ghosts game using machine learning, in: Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems - Volume 2, 2015, pp. 335–346. doi:10.1007/978-3-319-13356-0_27.

[14] S. Balakrishnan, K. Shunmuganathan, R. Sreenevasan, Amelioration of artificial intelligence using game techniques for an imperfect information board game Geister, International Journal of Applied Engineering Research 9 (22) (2014) 11849–11860.

[15] T. Mishio, Y. Kotani, Estimation algorithm UPP for imperfect information in games and application for Geister, Tech. rep., Tokyo University of Agriculture and Technology (2014).

[16] A. R. Buck, T. Banerjee, J. M. Keller, Evolving a fuzzy goal-driven strategy for the game of Geister: An exercise in teaching computational intelligence, in: 2014 IEEE Congress on Evolutionary Computation, 2014, pp. 28–35. doi:10.1109/CEC.2014.6900568.

[17] Y. Sato, Action-value function learning in Geister based on self-play games (in japanese), Master's thesis, The University of Electro-Communications, Chofu, Tokyo, Japan (2016).

[18] C. Chen, T. Kaneko, Utilizing history information in acquiring strategies for board game Geister by deep counterfactual regret minimization, in: Proceedings of the 24rd Game Programming Workshop 2019, 2019, pp. 20–27.

[19] K. Tomoda, K. Hasebe, Playing Geister by estimating hidden information with deep reinforcement learning, in: 2021 IEEE Conference on Games (CoG), 2021. doi:10.1109/cog52621.2021.9618992.

[20] B. De Kegel, M. Haahr, Procedural puzzle generation: A survey, IEEE Transactions on Games 12 (1) (2020) 21–40. doi:10.1109/TG.2019.2917792.

[21] M. Hirose, T. Ito, H. Matsubara, Automatic composition of tsume-shogi by reverse method, Journal of Japanese Society for Artificial Intelligence 13 (3) (1998) 452–460. doi:10.11517/jjsai.13.3_452.

[22] T. Mantere, J. Koljonen, Solving, rating and generating Sudoku puzzles with GA, in: 2007 IEEE Congress on Evolutionary Computation, 2007, pp. 1382–1389. doi:10.1109/CEC.2007.4424632.

[23] D. Oranchak, Evolutionary algorithm for generation of entertaining Shinro logic puzzles, in: Applications of Evolutionary Computation, 2010, pp. 181–190. doi:10.1007/978-3-642-12239-2_19.

[24] R. Takahashi, Mating problem generation of Puyo-Puyo for training, Master's thesis, Japan Advanced Institute of Science and Technology, Nomi, Ishikawa, Japan (2018).

[25] T. Oikawa, C.-H. Hsueh, K. Ikeda, Improving human players' T-spin skills in Tetris with procedural problem generation, in: The 16th Advances in Computer Games, 2019, pp. 41–52.

[26] G. d. B. de Trenquelleon, A. Choukarah, M. Roucairol, M. Addoum, T. Cazenave, Procedural generation of Rush Hour levels, in: The 11th International Conference on Computers and Games (CG 2022), 2023, pp. 181–190. doi:10.1007/978-3-031-34017-8_15.

[27] A. Iqbal, Evidence of the transmutation of creative elements using a computational creativity approach, in: 2022 IEEE 12th Symposium on Computer Applications & Industrial Electronics (ISCAIE), 2022. doi:10.1109/iscaie54458.2022.9794505.

[28] A. Kishimoto, M. H. Winands, M. Müller, J.-T. Saito, Game-tree search using proof numbers: The first twenty years, ICGA Journal 35 (3) (2012) 131–156. doi:10.3233/ICG-2012-35302.

[29] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 4th Edition, Pearson, 2020.

[30] C. Browne, Bitboard methods for games, ICGA Journal 37 (2) (2014) 67–84. doi:10.3233/ICG-2014-37202.

[31] A. Iqbal, H. van der Heijden, M. Guid, A. Makhmali, Evaluating the aesthetics of endgame studies: A computational model of human aesthetic perception, IEEE Transactions on Computational Intelligence and AI in Games 4 (3) (2012) 178–191. doi:10.1109/TCIAIG.2012.2192933.

[32] T. Ishitobi, Research of tsume-shogi composing algorithms, Master's thesis, Japan Advanced Institute of Science and Technology (2013).

[33] J. Togelius, G. N. Yannakakis, K. O. Stanley, C. Browne, Search-based procedural content generation: A taxonomy and survey, IEEE Transactions on Computational Intelligence and AI in Games 3 (3) (2011) 172–186. doi:10.1109/tciaig.2011.2148116.

[34] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgard, A. K. Hoover, A. Isaksen, A. Nealen, J. Togelius, Procedural content generation via machine learning (PCGML), IEEE Transactions on Games 10 (3) (2018) 257–270. doi:10.1109/tg.2018.2846639.