Technical Report

Lucas BESSON - Nathan BOSCHI - Robin HALM - Mathys NOURRY - Tom SIOUAN

SAE 1.2.6 - Développement IHM – Nine Men's Morris Game

Table of contents

Nine Men’s Morris presentation

Nine Men's Morris, also known as Merelle in French, is a timeless board game that has been enjoyed by people across the world for centuries. It is a two-player game that involves strategic placement and movement of pieces on a grid with the goal of leaving the opponent with fewer than three pieces, or blocking their moves so they cannot make any more.

The game is played on a grid with twenty-four points, also known as intersections, which are connected by lines forming a total of sixteen lines. Each player has nine pieces, usually in the form of discs or stones, and the game is played in three phases: placement, movement, and flying (alternative mode, not implemented). In the placement phase, players take turns placing their pieces on any empty point on the board until all the pieces have been placed. The objective is to form a mill, which is a row of three pieces along one of the lines. When a player forms a mill, they can remove one of their opponent's pieces from the board if the piece is not part of a mill. In the movement phase, players take turns moving one of their pieces to an adjacent point along one of the lines. The objective is to form new mills or to prevent the opponent from forming one by blocking their moves.

The game is won by leaving the opponent with fewer than three pieces, or by blocking their moves so they cannot make any more. The game can also end in a draw if neither player can make a legal move or if both players are left with only three pieces each.

In conclusion, Nine Men's Morris is a classic board game that has stood the test of time due to its simple yet strategic gameplay. With three (two implemented) distinct phases and the objective of leaving the opponent with fewer than three pieces, it is a game that requires careful planning and tactics to win.

Game Algorithms

First algorithm – Minimax algorithm

Presentation

Placing pieces

During the initial phase of the game, the algorithm checks which mills the player can complete. If there is a mill that can be completed (line of three pawns), the algorithm completes it by placing the third piece in the mill. If there are no mills that can be completed, the algorithm checks if the opponent can complete a mill. If so, the algorithm blocks the opponent's attempt by placing a piece in a location that disrupts the potential mill. If neither player can complete a mill, the algorithm places a piece randomly on the board.

Moving pieces

During the second phase of the game, the algorithm uses the minimax algorithm to determine the best move. The minimax algorithm considers all moves for the current player and the opponent, up to a certain depth limit. The algorithm evaluates each move by assigning a score to the resulting board position. The score is based on factors such as the number of pieces in mills, the number of potential mills, and the number of pieces that are threatened by the opponent. The algorithm then chooses the move that leads to the board position with the highest score. If the depth limit is reached, then victory is not close. The AI chooses a piece randomly and moves it.

Removing pieces

If a mill (line of three pieces) is formed, the AI looks for the piece that reduces the opponent's chances of winning. To do this, it chooses a piece that would allow the player to form a mill in the next turn. If the other player cannot form a mill in the next turn, the AI chooses a counter randomly and removes it.

Technical review

The minimax algorithm is a widely used technique for developing artificial intelligence in board games like Nine Men's Morris. In this algorithm, the computer evaluates all moves that it and its opponent can make and chooses the move that leads to the best outcome for itself. The algorithm works by recursively exploring the game tree, which represents all moves and their outcomes, and assigning a value to each node of the tree. The value of each node is determined by the minimax function, which alternates between maximizing and minimizing the node values depending on whose turn it is. This process continues until a predetermined depth or until a terminal node is reached, at which point the algorithm chooses the move that leads to the optimal outcome.

The performance of the minimax algorithm can vary depending on several factors such as the complexity of the game, the branching factor of the game tree, and the depth of the search. In terms of code volume, the algorithm can be implemented in a compact form, typically requiring a few hundred lines of code (here two hundred lines of code without utility methods). However, the memory usage can be quite significant, especially for games with large search spaces. The algorithm stores the game tree in memory, which can lead to memory overflow issues for deep search depths. To solve this problem, we have limited the search depth to avoid big and infinite sets.

The time required for the algorithm to make a move also depends on the game's complexity and search depth. For Nine Men's Morris, the average time required for the algorithm to make a move can range from a few milliseconds to several seconds, status of the game and the available computational resources. In practice, minimizing the search depth is a common way to speed up the algorithm's execution time. This approach sacrifices accuracy for speed, but it can be effective for games that require quick decisions.

Overall, the performance of the minimax algorithm can be optimized by carefully tuning the search depth and by implementing various optimization techniques such as alpha-beta pruning, transposition tables, and iterative deepening. These optimizations can significantly improve the algorithm's performance while reducing the memory usage and execution time.

Performance evaluation

To evaluate the effectiveness of the minimax algorithm, we assessed it against several other playing strategies, including a random player and a human player. The minimax algorithm performed well, winning a substantial number of games against all opponents, and holding its own against the human player.

The algorithm's performance was particularly strong in the mid-to-late game, where its ability to look ahead and evaluate multiple moves gave it a strategic advantage. However, in the early game, where fewer pieces are on the board and the game tree is less complex, the minimax algorithm was not significantly better than other strategies.

Second algorithm

Presentation

The algorithm consists of two main phases: placement and movement. In the placement phase, the algorithm examines the board to see if it can complete a mill. If it can, it places a piece in the third position to complete the mill and remove one of the opponent's pieces. If it cannot complete a mill, it places a piece randomly on the board.

In the movement phase, the algorithm examines all the player's pieces to see if any can form a mill. If it finds a mill, it completes it and removes one of the opponent's pieces. If it cannot find a mill, it selects a random piece that can be moved and a random position to move it to.

Lucas BESSON - Nathan BOSCHI - Robin HALM - Mathys NOURRY - Tom SIOUAN

Etudiants en première année de BUT Informatique

Université de Franche-Comté
IUT Nord Franche-Comté
19 Av. du Maréchal Juin
90016 Belfort

Technical review

The placement phase is designed to prioritize completing mills, which are known to be strong strategic moves in the game. By focusing on mills, the algorithm can make progress towards winning the game even if it must make random moves in between.

The movement phase is designed to balance exploration and exploitation. By examining all the player's pieces, the algorithm can identify opportunities to form mills and take advantage of them. However, if no mills are available, the algorithm selects a random piece to move to avoid getting stuck in local optima.

The use of randomness in the algorithm is critical to its effectiveness. By randomly selecting moves, the algorithm can explore the game tree more fully and avoid getting stuck in suboptimal paths. However, the randomness is tempered by the focus on completing mills, which are known to be strong strategic moves.

In conclusion, the randomized algorithm presented in this report is an effective way to play Nine Men's Morris. By balancing exploration and exploitation and focusing on strong strategic moves like completing mills, the algorithm can hold its own against more complex strategies and even human players. However, its effectiveness is limited by the game's inherent complexity and the diminishing number of available moves as the game progresses. Overall, this algorithm represents a significant step forward in the development of artificial intelligence for playing Nine Men's Morris.

Performance evaluation

The algorithm's performance was particularly strong in the early game, where its focus on completing mills gave it a strategic advantage. However, as the game progressed and fewer pieces were available, the algorithm's randomness became more of a liability, and it struggled to find strong moves.

Entry files

Entry files allow the development team to evaluate the game while developing, which can help us to identify missing functionalities and detect bugs rapidly.

Here is three entry file that has been used during the development of Nine Men's Morris, which are explicitly name based on their purpose:

InputPlayerWinAfterMills.txt

Entry file that allows us to evaluate both phases of the game, placing and moving, with mills action, which make the player one win by deleting all the opponent pawns.

InputPlayerWinAfterBlockingOponent.txt

Entry file that makes the player one win by blocking the opponent pawn with the deletion of two pawns.

InputEndingGameWithEdgerCaseTest.txt

Entry file that allows us to evaluate the edge case (impossible move and programming flaws) and should make the player one win if all the edge case are managed correctly.

Lucas BESSON - Nathan BOSCHI - Robin HALM - Mathys NOURRY - Tom SIOUAN

Etudiants en première année de BUT Informatique

Université de Franche-Comté
IUT Nord Franche-Comté
19 Av. du Maréchal Juin
90016 Belfort