EXIMO

Report – 2020-03-31

Bernardo Santos - 201706534
João Nuno Matos - 201705471
Vítor Gonçalves - 201703917

Eximo

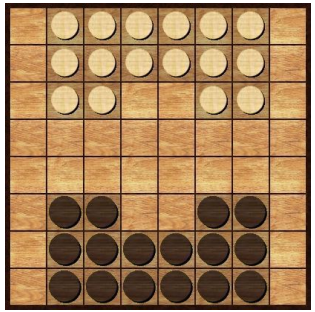

Figure 1: Initial board

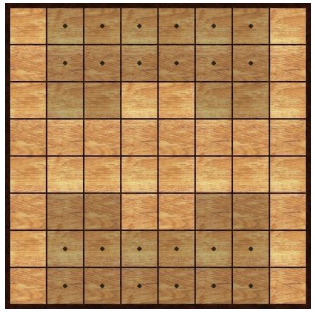

Figure 2: Drop zones

Specification

Goals:

- capture all opponent's pieces
- stalemate the opponent, so he has no moves left

Turn - in each turn a player can make one of two actions:

- **move:** a piece can move in three directions forward or diagonally forward
  - **ordinary move:** move to an adjacent and empty square (Figure 3)
  - **jumping move:** jump over an adjacent friendly piece if the next square in the same direction is empty, placing the piece on the empty square. If the same player's piece can continue moving by jumping another friendly piece, then it must do so (Figure 4). During the jumping move that piece cannot capture enemy pieces.
- **capture**: a piece jumps over a (forward, diagonally forward, right or left) adjacent enemy's piece if the next square in the same direction is empty, placing the piece on that same square and removing the opponent's piece from the board (Figure 5). If the piece can continue capturing, then it must do so (capturing is mandatory, and the player must keep capturing as long as possible).

When a piece reaches the opposite side of the board:

1. The piece is removed
2. Two pieces are placed on the player's drop zone (Figure 2)
   a. if there are no empty cells, the player loses the piece
   b. if there is only one empty cell, the player only places one piece

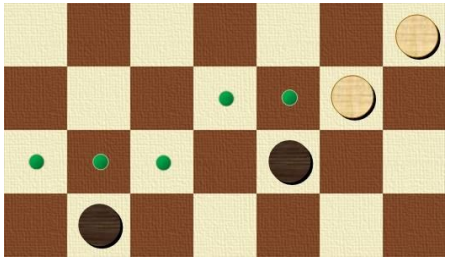

Figure 3: Ordinary move

Figure 4: Jumping move

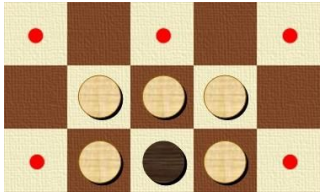

Figure 5: Capture

Formulating *Eximo* as a Search Problem

- State features:
  - Board (8x8 integer matrix)
  - Whose turn it is
  - Stage of the turn (start of turn/jumping over allies/capturing enemies/placing new pieces)

  Initial state is board = board in Fig. 1, player = 1, state = *Start*.
- Operators
  - Move operator: advances one piece one cell forward (left, right, or center).
    - Must be at *Start* state;
    - An adjacent cell must be empty
    - Must not be able to capture a piece
    - Move piece; Switch current player;
  - Jump operator: advances one piece two cells forward (left, right, or center) over an ally piece.
    - Must be at *Start* or *Jump* state (if in *Jump* state, jumping piece must be the tagged one);
    - An adjacent cell must contain an ally, and the next cell in that direction must be empty
    - Must not be able to capture a piece
    - Move piece; If piece can jump again go to *Jump*, else switch player and go to *Start*

Formulating *Eximo* as a Search Problem (cont.)

- Capture operator: advances one piece two cells forward (left, right, or center) or to the side (left or right) over an enemy piece, removing the enemy piece.
  - Must be at *Start* or *Capture* state (if in *Capture* state, jumping piece must be the tagged one)
  - An adjacent cell must contain an enemy, and the next cell in that direction must be empty
  - Move piece, remove enemy. If can capture again go to *Capture*, else go to *Start* and switch player.

  In Move, Jump and Capture, if piece reaches far end of board, remove it and go to Place
- Place operator: places one piece, if possible, in the landing area of the player.
  - Must be at *Place* state
  - If done placing, go to *Start* and switch player, else continue placing.

- Goal state is one of either:
  - Losing player has no pieces left at the start of their turn
  - Losing player cannot do any more moves (cannot fulfill any operator's preconditions)

Implementation Approach

- Implementation Environment
  - Python 3 (mainly on Linux). Uses the Colorama library to display colour in the terminal UI.
  - File structure:
    - main.py - application entry point
    - eximo.py - core game logic and search algorithms
    - state.py - “State” class: data structure and operators
    - eval.py - heuristic functions
    - aux.py - miscellaneous helper classes/functions; tests.py - facilities for manual testing
- Data structures
  - State: comprised of following fields
    - board - list of integer lists
    - player - integer, either 1 or 2
    - action - tagged union ([tag, field]) of different stages of the turn
      - field may not exist (Start), be the position of a piece (Jump, Capture), or the number of remaining pieces to place (Place)
    - score - dictionary that maps player → player’s “score“

Implementation Approach (cont.)

    - Position and direction (tuples) - along with addition and multiplication operators, useful for quickly applying operators.
- Implemented Heuristics
    - Maximizing number of pieces
    - Maximizing difference between our and our opponent’s number of pieces
    - Prioritize moves to the side columns of the board
    - Prioritize moves to the center rows of the board
- Search Algorithms
    - Minimax Adversarial Search
        - Configurable regarding depth (1 is greedy search)
        - Configurable regarding heuristics used
        - Implemented in Eximo.minimax
    - Minimax Adversarial Search with Alpha-Beta-Pruning
        - Configurable regarding depth (1 is greedy search)
        - Configurable regarding heuristics used
        - Will cut off branches when it is sure parent would not choose it
        - Implemented in Eximo.minimax_prunning

Experimental Results

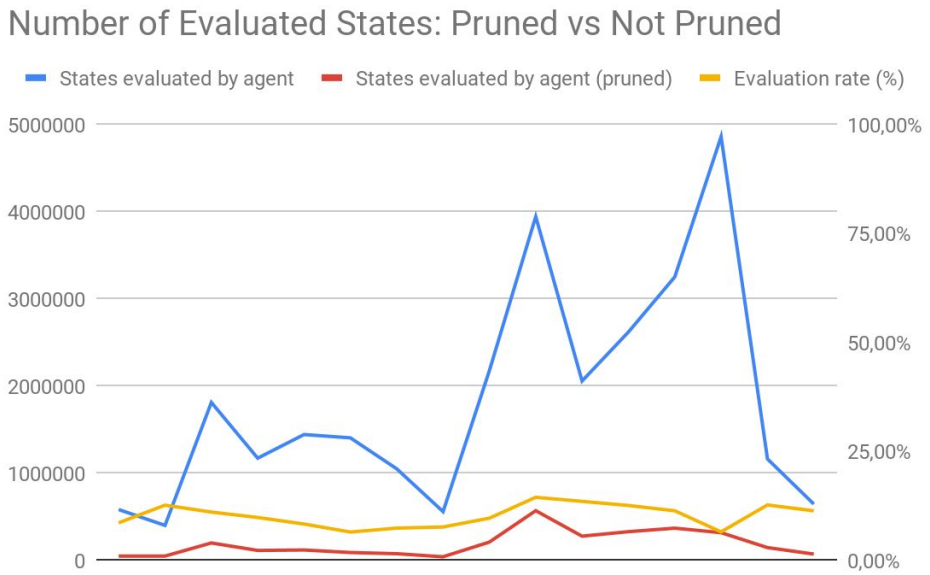

Figure 6: Comparison between pruned and regular minimax adversarial search. We observed savings of one order of magnitude (~90%).

Conclusions

- A reasonably intelligent mechanism to develop an agent that can play games can be achieved by modelling the game as a search problem.
- The optimal solution to an adversarial search problem can be approximated by performing a depth-limited minimax adversarial search.
- The choice of heuristics is of extreme importance in achieving good results
- Pruning the search tree to remove unviable branches effects in substantial performance gains

References

- exposure and rules of the game
  - https://www.boardgamegeek.com/boardgame/137916/eximo
- heuristics file
  - https://www.mini.pw.edu.pl/~mandziuk/PRACE/es_init.pdf
- module used for color usage
  - https://pypi.org/project/colorama/