Artificial Intelligence A2 - Games

Franco Ruggeri
Javier García San Vicente

Q1: Describe initial state, possible states and transition function in checkers

- Initial state: 12 checkers of each color positioned in opposite sides of the board.
- Possible states: every possible disposition of k checkers in the black squares of the board, with 1 <= k <= 12.
- Transition function: a piece can
  - Move in diagonal by one position forwards or, only if it is king, backwards
  - Jump one opponent's piece forwards, only if it is in an adjacent square and the following square is empty. If after the first jump another jump is possible, it must be done in the same move. In case of kings, jumps can be done also backwards.

Q2: Describe the terminal states of both checkers and tic-tac-toe.

- Checkers:
  - All pieces of the opponent captured (the current player wins).
  - No moves possible for the current player (the opponent wins).
  - 50 moves (i.e. 25 for each player) with no captures (draw).
- Tic-tac-toe:
  - 4 marks of the same player in the same row (the player wins).
  - No possible rows of 4 marks of the same player (draw).

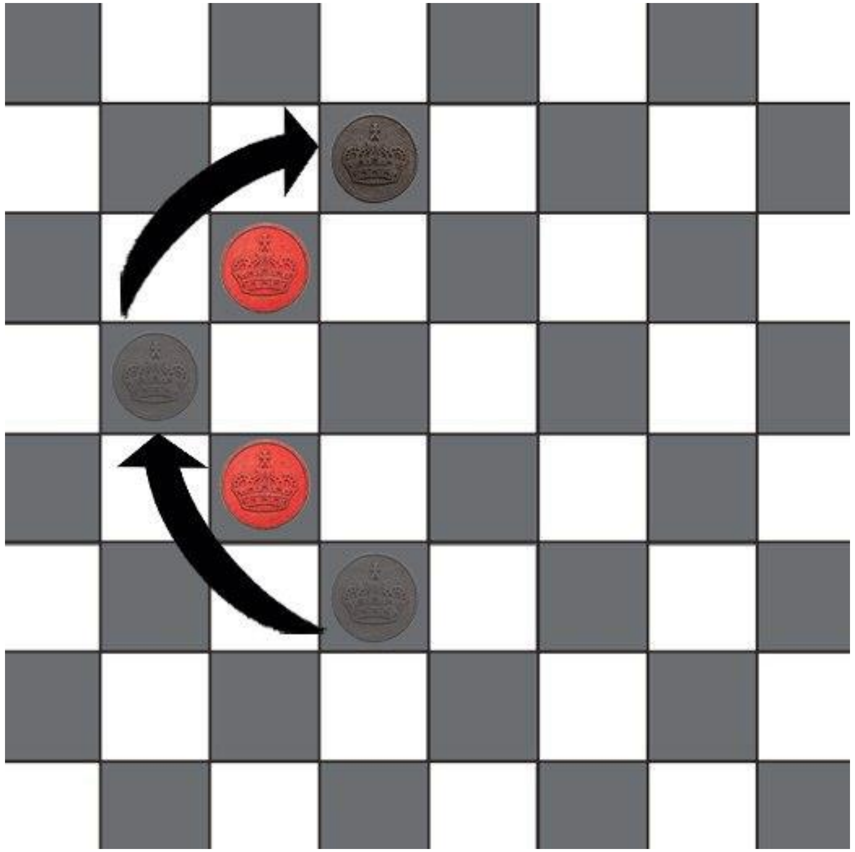
Q3: Why is v(A,s) = #{white checkers} - #{red checkers} a valid heuristic function for checkers?

- It orders the terminal states the same as the utility function
  - win → $v(A,s) > 0$
  - loss → $v(A,s) < 0$
  - draw → $v(A,s) = 0$
- It is fast to compute
- It reflects (more or less, see next slides) chances to win

Q4: When does v(A,s) best approximate the utility function, and why?

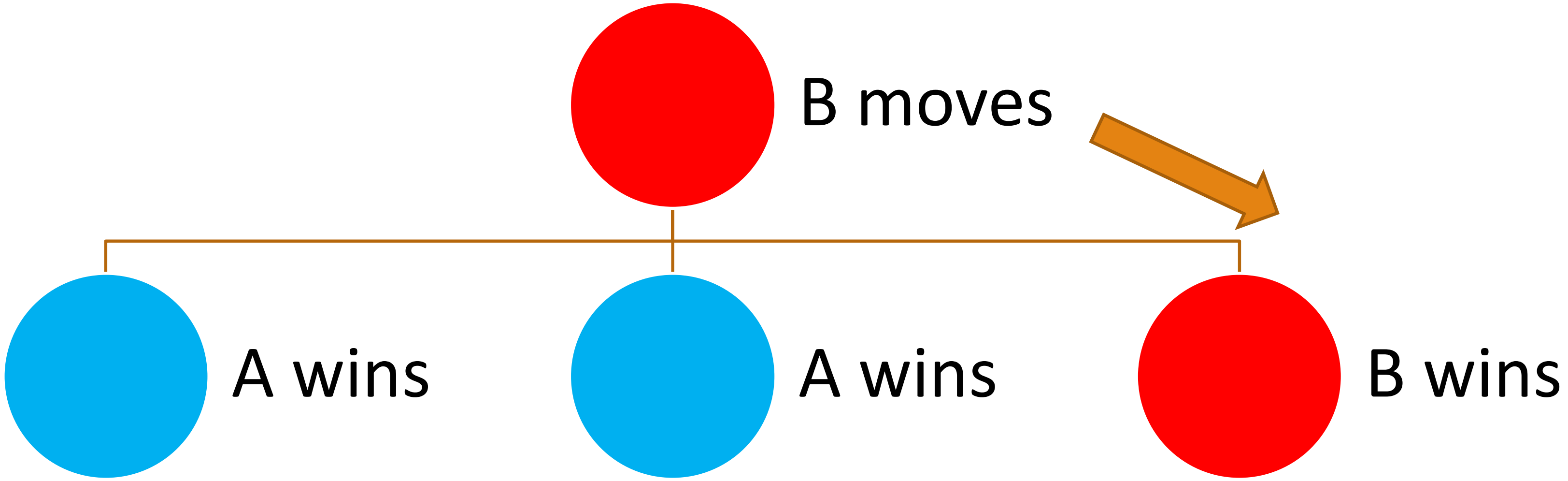
- The best approximation is in the terminal states (see the previous slide).
- When kings are involved, v(A,s) is of course not good, since kings have more impact on the game than normal pieces.

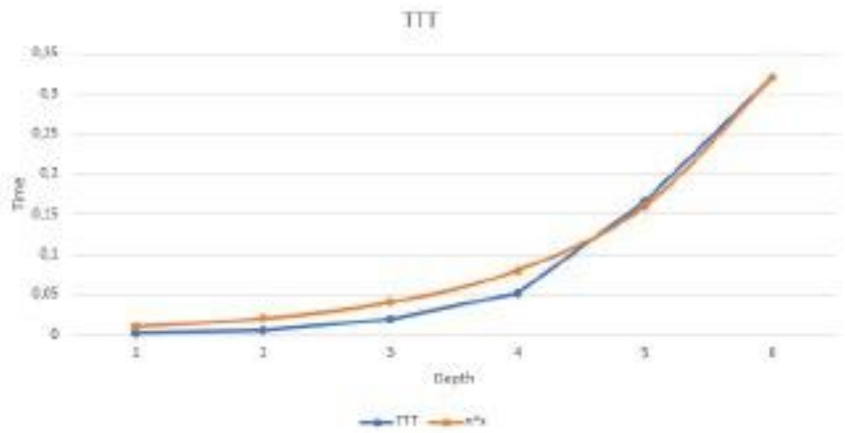
Q5: Can you provide an example of a state s where v(A,s) > 0 and B wins in the following turn?

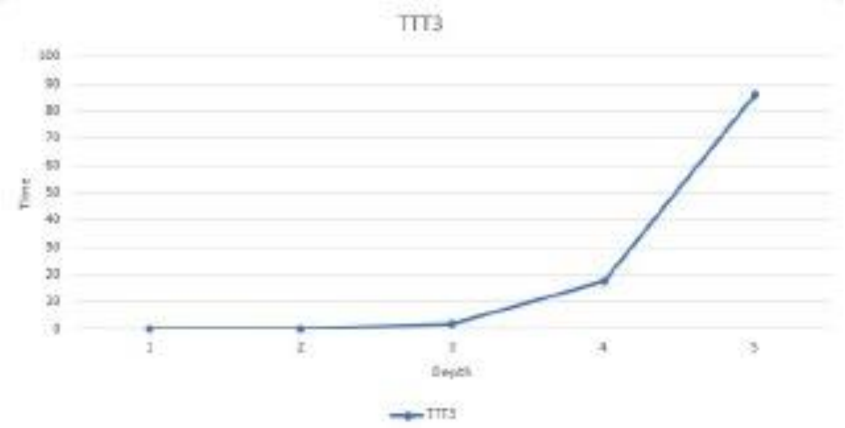


Q6: will n suffer from the same problem as the evaluation function v?

- Yes, because even if the majority of terminal states are win of A, B can choose a winning state for him.







Tic-tac-toe: impact of depth

- Complexity of minimax truncated at depth d: O(b^d)
- The execution time increases exponentially with the depth.
- Alpha-beta pruning helps to avoid visiting parts of the tree, but depth has also heavy impact.
- Possible optimizations to increase pruning:
  - Repeated states checking
  - Symmetry breaking
  - Move ordering

Checkers: optimizations

- Iterative deepening search
- Move ordering
  - Idea: let’s order the possible moves such that those with high impact are tried first.
  - Order:
    1. Jumps, sorted by decreasing length
    2. Normal moves resulting in a piece being crowned
    3. Rest of the normal moves
- Repeated states checking
- Symmetry breaking