Advanced AI Agent for the Game Hex

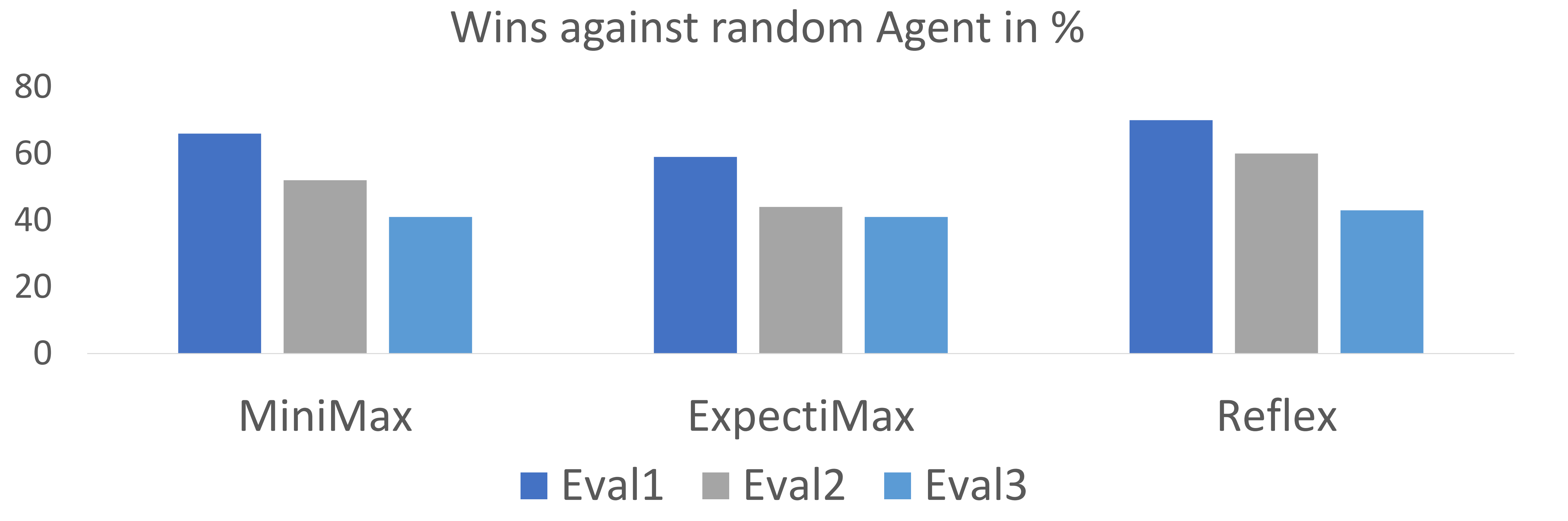
Axel Gross-Klussmann, agk1982@Stanford.edu. Video: https://drive.google.com/file/d/12KpoO5Fm9aTOAMHMD3yFfu0fNHyBsweq/view

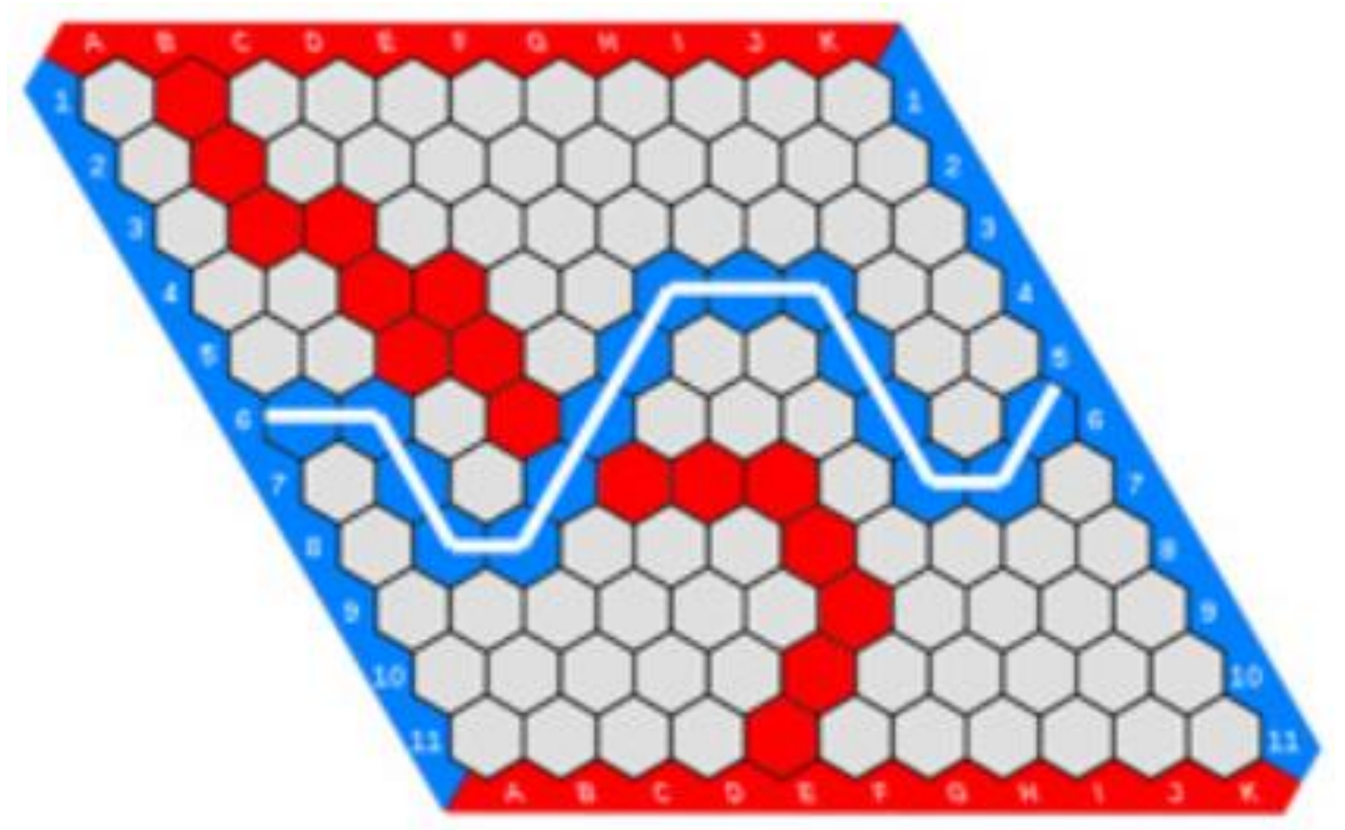
Overview and Background

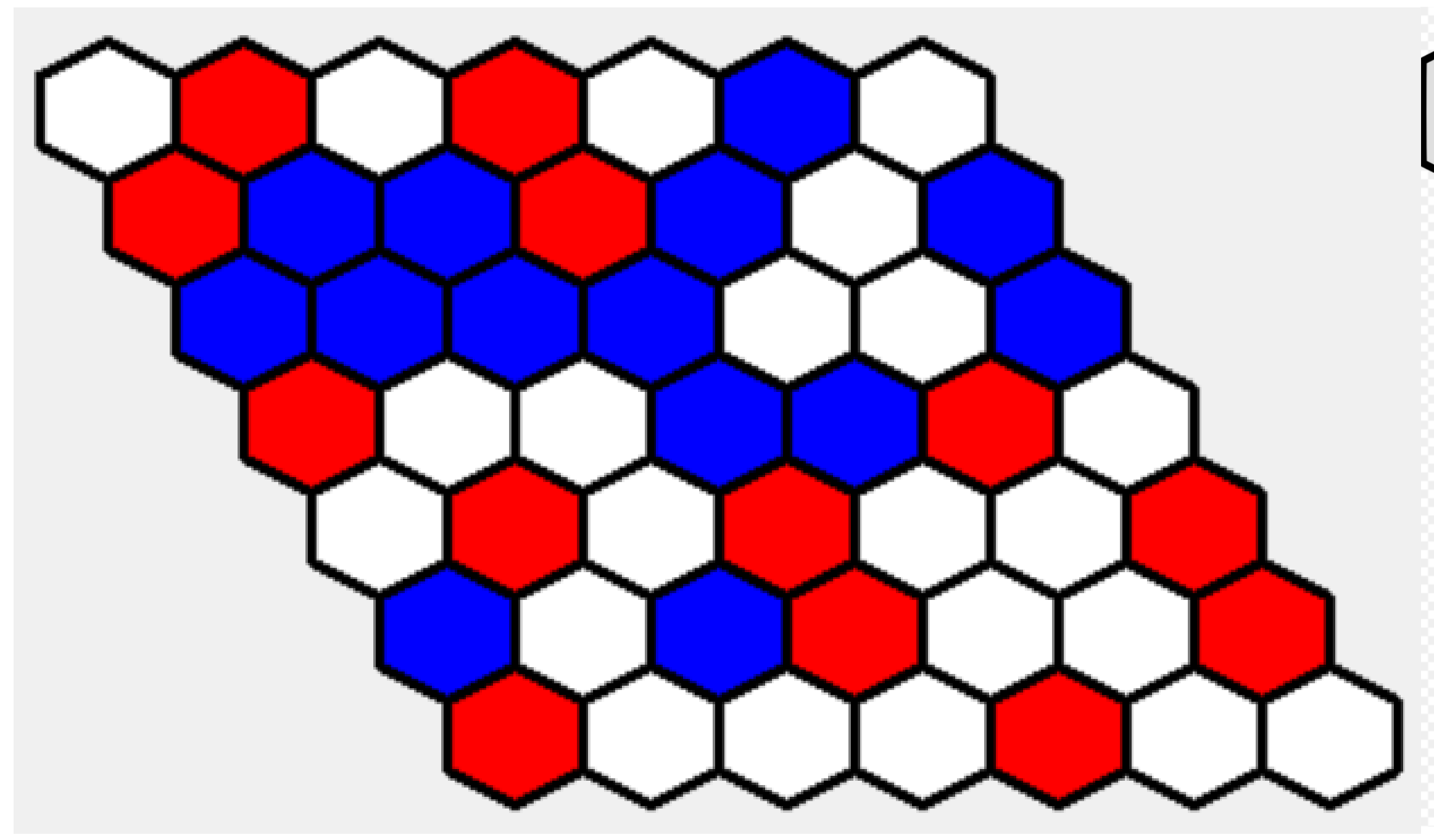
- Hex is a 2 player zero sum board game
- Co-invented by the famous Mathematician John Nash
- Played on a hexagonal grid, typically an 11x11 rhombus
- Game has deep strategy, interesting theoretical properties and can never end in a draw

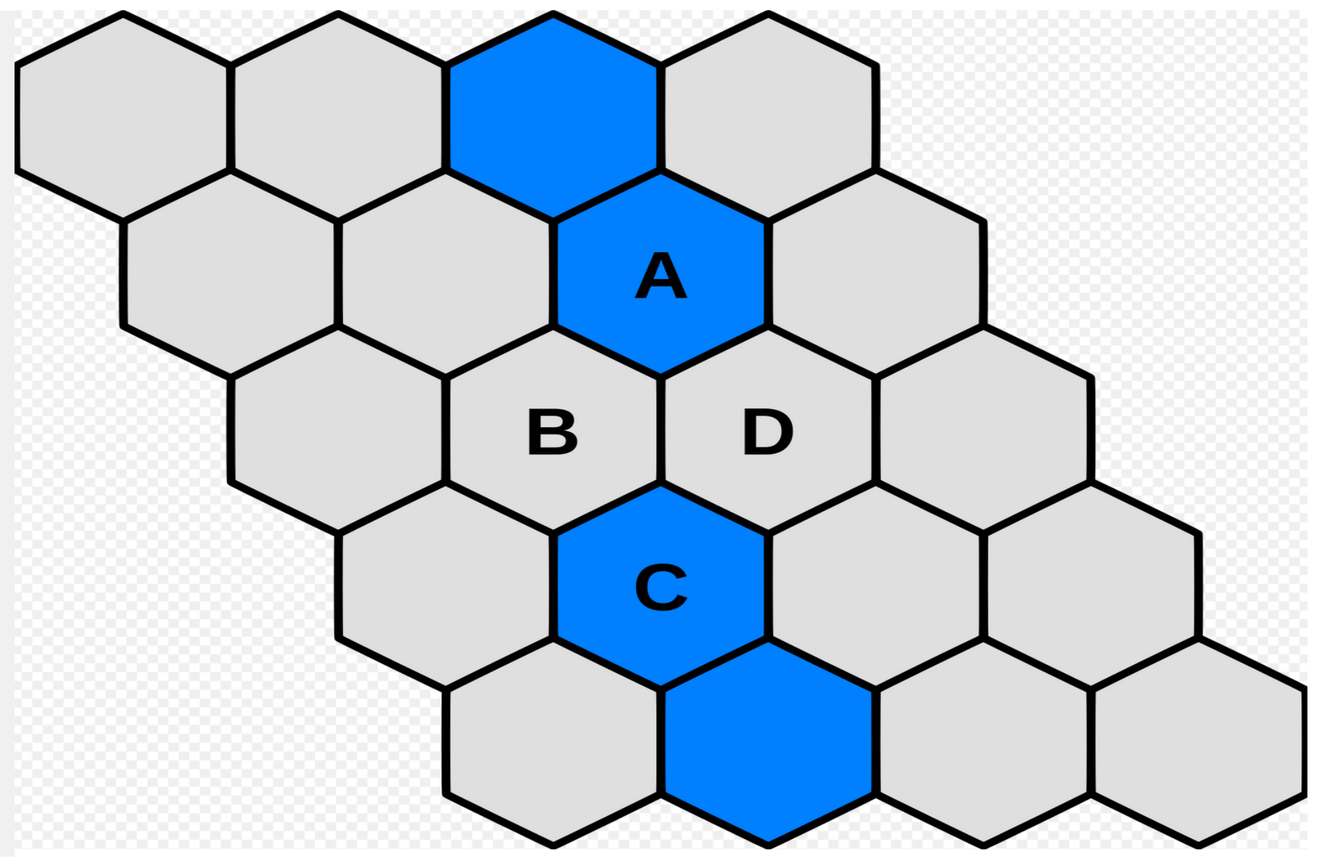
Modeling Approach & Challenges

- The main challenge in modeling Hex is the computational complexity of the board which by far surpasses the complexity of chess(!)
- Our AI agent relies on a state space approach
- Exploring the full game tree is infeasible and forward-looking AI players have to rely on heuristics evaluating the (future) game states
- We model the hexagons as graph which allows to use UCS / A* search









Experiments

- We test three AI agents (MiniMax, ExpectiMax and Reflex) against a random player (the baseline) who draws uniformly from the grid
- Further, we enhance the eval function by accounting for board coherence (see the middle picture) as well as the „bridge“ strategy (right picture)
- Eval1 denotes the mentioned path cost evaluation,
- Eval2 represents a coherence measure and Eval3 accounts for the bridge strategy
- The above stack plot shows %-wins per agent and eval function for 1000 runs on a 4x4 board

Rules

- Luckily, the rules are quite simple
- Two competing players try to connect their „home“ sides by occupying empty fields of the hex board in turns
- The above board exemplifies a situation where player blue has to connect the left to the right and the red player‘s aim is to connect top to bottom side
- Traversing across fields can use all six sides of the hexagons
- Classical strategy: Block paths

Evaluating Game States

- We employ the limited depth Minimax framework for our main agent:

$$V_{minmax}(s,d) = \begin{cases} Utility(s) & \text{IsEnd(s)} \\ Eval(s) & d=0 \\ \max_{a\in Actions(s)} V_{minmax}(Succ(s,a),d) & \text{Player(s)} = \text{agent} \\ \min_{a\in Actions(s)} V_{minmax}(Succ(s,a),d-1) & \text{Player(s)} = \text{opp} \end{cases}$$

- We use UCS to determine whether the sides of a player are connected and collect the path costs (isEnd(s) has cost 0)
- We experiment with several eval functions. The simplest case collects the costs for traversing from own (cost 0) and adversarial fields (cost 1) to the other side and sets the utility as 1/costs

Discussion of Results and Future Work

- While the minimum cost path evaluation function beats the random agent in more than 50% of cases, the dismal performance of the „coherence“ and „bridge“ eval functions shows that these are insufficient in their current form
- Further, the reflex agent performs strongest

References

Chalup, Mellor and Rosamond (2005), The machine intelligence hex project, Computer Science Education, 245-273