The Second NeurIPS Tournament of Reconnaissance Blind Chess

Gino Perrotta GINO.PERROTTA@JHUAPL.EDU
Ryan W. Gardner RYAN.GARDNER@JHUAPL.EDU
Corey Lowman COREY.LOWMAN@JHUAPL.EDU
Johns Hopkins University Applied Physics Laboratory
Mohammad Taufeeque TAUFEEQUE@CSE.IITB.AC.IN
Nitish Tongia NITISHTONGIA@EE.IITB.AC.IN
Shivaram Kalyanakrishnan SHIVARAM@CSE.IITB.AC.IN
Indian Institute of Technology Bombay
Gregory Clark GREGORYCLARK@GOOGLE.COM
ML Collective
Google
Kevin Wang KEVINWANG@KEVINWANG.US
Independent
Eitan Rothberg ROTHBERG.10@BUCKEYEMAIL.OSU.EDU
Independent
Brady P. Garrison BRADYPGARRISON@GMAIL.COM
Prithviraj Dasgupta RAJ.DASGUPTA@NRL.NAVY.MIL
U.S. Naval Research Laboratory
Callum Canavan CALLUM.CANAVAN@SEAGATE.COM
Seagate Technology
Lucas McCabe LMCCABE@LMI.ORG
LMI

Editors: Douwe Kiela, Marco Ciccone, Barbara Caputo

Abstract

Reconnaissance Blind Chess is an imperfect-information variant of chess with significant private information that challenges state-of-the-art algorithms. The Johns Hopkins University Applied Physics Laboratory and several organizing partners held the second NeurIPS machine Reconnaissance Blind Chess competition in 2021. 18 bots competed in 9,180 games, revealing a dominant champion with 91% wins. The top four bots in the tournament matched or exceeded the performance of the inaugural tournament's winner. However, none of the algorithms converge to an optimal, unexploitable strategy or appear to have addressed the core research challenges associated with Reconnaissance Blind Chess.

Keywords: reconnaissance blind chess, imperfect information, reinforcement learning, common knowledge

1. Introduction

A multitude of games have historically provided clearly defined, strategically complex environments that are useful for studying and comparing automated decision-making algorithms. Advancements in such algorithms have been marked by the achievement of superhuman play in various games, famously including backgammon (TD-Gammon) (Tesauro, 1995), chess (Deep Blue, AlphaZero) (Campbell et al., 2002; Silver et al., 2018), Go (AlphaGo, AlphaZero) (Silver et al., 2017b), shogi (AlphaZero), and Texas hold-em poker (Libratus, DeepStack) (Brown and Sandholm, 2017; Moravčík et al., 2017).

Games with significant private information remain a challenge for existing algorithms. However, games of this nature are scarce because such games usually require a referee to pass information and validate actions. To support continuing developments in automated decision making under uncertainty, researchers at the Johns Hopkins University Applied Physics Laboratory (JHU/APL) created the game Reconnaissance Blind Chess (RBC) as a proposed common benchmark. RBC is a blind chess variant in which observations are primarily private, designed to reflect the challenges of automated decision making in non-game (or "real world") strategic scenarios. JHU/APL hosted an inaugural competition of RBC algorithms as part of the Neural Information Processing Systems (NeurIPS) conference in 2019, and hosted a second NeurIPS tournament in 2021.

This paper describes the rules and research purpose of the game RBC and of the NeurIPS 2021 tournament. Participants in the tournament provide summaries of their algorithms, and tournament organizers review post-competition analysis and lessons learned. Although four participating agents appear stronger than 2019's winning algorithm, none converge to an optimal strategy or provide a comprehensive, practical solution to the game's research challenges. RBC remains an open and promising research problem.

2. Reconnaissance Blind Chess

RBC is an imperfect information variant of chess (Newman et al., 2016). Like other blind chess variants, RBC requires that players be separated and communicate through a referee. It is best played on a computer. An RBC player cannot in general see her opponent's pieces. Instead, their possible positions are deduced from:

1. The known starting arrangement, which is the same as in chess.

2. A sensing action taken on each turn before moving. The referee informs the sensing player of the contents of a requested 3×3 square area on the chess board. A player is never informed where her opponent sensed.

3. The location of captures, provided by the referee. The type of the opponent's piece which was captured or did the capturing is not reported.

4. Corrections to a player's own illegal moves are also provided by the referee. (Corrections to her opponent's moves remain hidden.) Moving a sliding piece through an unseen opposing piece results in the moved piece stopping and making a capture. Illegal castling or pawn moves result in no move at all, except when advancing a pawn two squares, which can result in a single step if only the second square is occupied.

This dedicated, private sensing action is unique to RBC, and gives it the characteristics that make RBC an interesting platform for research. The privacy of this observation makes it difficult to reason about an opponent's knowledge, while the power of the observation makes it essential to do so. Depending on the senses made, an opponent may have perfect information about the state of the chess board or nearly total uncertainty; a player's optimal actions could be quite different between these circumstances.

The chess rules regarding check are removed from RBC. Instead, a player wins by capturing the opponent's king. For this competition, a chess clock was also used. Either player loses the game after exhausting 15 minutes of cumulative time. An interested reader can learn more and play the game at `https://rbc.jhuapl.edu`.

3. Research Challenges and Related Work

Algorithms for playing perfect-information games take advantage of natural properties of such games: 1, the game state is always known exactly, so computing a strategy mid-game only requires searching in a subgame beginning from the current state, and 2, an optimal strategy can be deterministic, as there is no incentive (or ability) to play unpredictably. Algorithms such as minimax and Monte Carlo tree search (MCTS) (Kocsis and Szepesvári, 2006; Browne et al., 2012) can efficiently identify a sequence of actions that result in the best reachable state from the acting player's perspective at each decision point. In complex games with larger state spaces, like chess and Go, an optimal strategy can only be approximated, and research efforts focus on limiting the search computation, often by creating functions that can accurately estimate the value of a game state without needing to play to a terminal state (Silver et al., 2017b, 2018).

In general, however, computing optimal actions in a game of *imperfect* information is far more difficult. Because the exact state of the game is not known, optimal actions are probabilistic, with probabilities that depend on the likelihood of the currently-possible game states and the opponent's following actions. This is true of both players, so estimation of an opponent's policy reflects the probabilities of currently-possible game states *from the opponent's perspective* and the opponent's estimation of the original player's policy.[1] All possible states in a game of imperfect information can affect the optimal actions at any given state. This dramatically increases the computational cost of finding optimal actions compared to games of perfect information, even for games with similar numbers of states.

Despite this, there have been breakthroughs in games of imperfect information including superhuman play in no-limit Texas hold-em poker (Brown and Sandholm, 2017; Moravčík et al., 2017). Algorithms like counterfactual regret minimization (CFR) (Zinkevich et al., 2008) model all players and provably converge to an optimal (Nash-equilibrium) strategy. This requires repeatedly simulating games *from the beginning*, which can create an intractably-large search space in many games. Several approaches have been developed to focus this search within subgames. Some approaches limit the search breadth by dividing the game into public belief states—sets of game states grouped by information available to *all* players. Online search is restricted to the current public subgame, using saved information to represent the essential portions of the rest of the game (Burch et al., 2014; Moravčík et al., 2017; Sustr et al., 2018; Brown et al., 2020). Other approaches limit search depth by learning and storing public state values in a neural network (Moravčík et al., 2017; Brown et al., 2020). These techniques can yield strong play for Texas hold-em poker, in which the only non-public information is each player's private hand. In RBC, however, there is much

1. Furthermore, this applies recursively. The current optimal actions depend on a player's estimate of the opponent's estimate of the player's estimate... and so on.

more private information, and thus public subgames often include a tremendous number of states. A different approach restricts search depth by choosing actions beyond the depth limit using fixed strategies for all players, but permitting a choice from among several fixed strategies for opponents (Brown et al., 2018). A method for practically computing these strategies is not known for a game as large and complex as RBC.

Alternately, search can be foregone in favor of approximating equilibrium policies for the whole game through deep-CFR (Brown et al., 2019) or neural fictitious self-play (NFSP) (Heinrich and Silver, 2016). In large, complex games, training the necessary neural networks could be infeasible. Additionally, online strategy search has been shown to be invaluable to strong game play (Silver et al., 2017a).

Another adaptation to CFR, online outcome sampling (OOS) (Lisý et al., 2015), importance samples game playouts that are consistent with a player's current information. This is not sufficient for practical computation of an RBC strategy for several reasons, including that every possible sequence of opponent sense actions is always consistent with information available to a player, leaving an exponentially-increasing number of states to be importance sampled.

Zhang and Sandholm (2021) developed knowledge-limited subgame solving (KLSS) specifically to address games with little common knowledge. Their algorithm restricts a subgame to states that are possible within a certain knowledge distance, where the games states consistent with the acting player's information have a distance of 0, the states consistent with any opponent information which itself is consistent with the player's information have a distance of 1, and so on. KLSS limited to a knowledge distance of 0 (called 1-KLSS) was used to play dark chess, although not at superhuman level. As with OOS, RBC's exponential growth in number of states (even at distance 0) may challenge KLSS.[2]

Literature on imperfect-information games includes other blind chess variants, Kriegspiel (Ciancarini and Favini, 2010; Russell and Wolfe, 2005) and dark chess (Zhang and Sandholm, 2021). In Kriegspiel, pieces are never directly observed. A player learns about their opponent's pieces by attempting moves, which are rejected if illegal (the player then tries again). Players are also informed of checks and captures. In dark chess, squares are visible to a player if they have a legal move which ends on that square. The concept of check is removed and a game is won by capturing the opponent's king. Both Kriegspiel and dark chess couple observations to piece placement, which reduces the amount of private information compared to RBC, although each presents similar practical challenges to search algorithms.

4. Competition Structure

The NeurIPS 2021 RBC competition followed the format used in the inaugural 2019 competition (Gardner et al., 2020). The event was open to any participants interested in writing an RBC bot. During the tournament, bot authors ran their own code, which communicated online with the RBC server. The server scheduled, refereed, and recorded the games. The competition was a *multi-round-robin* tournament; every participant played against each op-

2. Zhang and Sandholm (2021) partially addressed this by considering game states which are *from that moment on indistinguishable to an outside observer* to be transpositions—effectively the same state. In dark chess, this disregards only move order, which may mean that transpositions are practically similar. In RBC, however, this also disregards the history of sense actions.

ponent an equal number of times. This tournament held 30 rounds, and in every round all bots faced each opponent twice, once on each side of the chess board.

The rank for each bot was decided using Bayesian Elo rating[3] (Coulom, 2006). This estimate of relative player strength is particularly well suited for round robin tournaments; the results are invariant to game order, but sensitive to the number of matches between each pair of participants, which here was constant.

The primary organizer, the Johns Hopkins University Applied Physics Laboratory, offered a $1,000 prize to the winner and $500 to the runner-up, excluding bots written by affiliates of Johns Hopkins University. Bots were limited to 15 minutes of cumulative time per bot per game, and were required to support four simultaneous games. The tournament took 72 hours to complete, starting on October 20, 2021.

5. Overview of Approaches

This section provides a brief summary of most of the participating bots in the NeurIPS 2021 tournament of RBC. Table 1 contains algorithm descriptions ordered by descending tournament rank. Table 2 compares submitted agents by high-level features included in or excluded from their algorithms. Most bots perform exhaustive tracking of possible states of the chess board, but only four track those possibilities from the opponent's point of view, and none track the recursive belief states that form the full RBC game state. Top performing bots tend to predict their opponent's moves, which may be made more reliable by the prevalence of chess engines for move evaluation. Several bots choose sense actions to minimize the expected number of remaining states, but most of the top performers instead choose sense actions with a more general game objective in mind.

Table 1: Brief description of select bots' algorithms.

Bot Name	**Brief Description**
Fianchetto	Built using the source code of StrangeFish (Perrotta and Perrotta, 2019), in which it introduces several major changes. For board evaluations, Fianchetto uses Lc0 (Pascutto and Linscott, 2018), rather than Stockfish (Romstad et al., 2018). Lc0 speeds up the evaluation step by giving scores to all possible actions through a single forward pass of a neural network, with the additional option of evaluating multiple boards in parallel on a GPU. The gain in speed enables the expansion of the search tree to include a probabilistic model of opponent behavior, itself also taken from Lc0. Fianchetto maintains a belief over all possible board states, which is recomputed on each turn using the POMDP belief update equation.
StrangeFish2	Extends the exhaustive board state tracking and average outcome move selection of its predecessor, StrangeFish. The sense actions are selected by estimating probabilities for all possible observations, choosing a move following each of those hypothetical observations, and estimating probabilities for all possible outcomes of that move. These probabilities are based on the estimated distribution over board states, which is a function of the opponent's strength in each position. The choice of sense is then that which leads to, on average, the best outcome after this turn's move. Move and state values are computed using Stockfish plus uncertainty related heuristics.

3. In this competition, the Elo estimate was computed with zero advantage assigned to the first-to-move player, and zero likelihood of a draw (drawing a game was not possible in RBC as implemented at the time). Bayesian Elo produces an Elo estimate with confidence intervals for each player. The central estimate was used as each player's score, without adjusting towards to lower confidence bound.

Bot Name	**Brief Description**
penumbra	Based on deep synoptic Monte Carlo planning (Clark, 2021). It tracks all possible opponent board states, maintains a belief state with an unweighted particle filter, and plans with upper confidence bound tree search. It approximates information states with a stochastic abstraction, encoding sets of boards with compact fixed-size synopses. Those synopses are provided as input to a deep neural network to guide the search algorithm. The 12-block residual network was trained to model twenty opponents based on historical game data.
Kevin	Explicitly keeps track of all possible board states and a probability distribution over them. It chooses sense actions to minimize the expected difference in value between its next move and its theoretical best move for the true state. Each opponent turn, it calculates the probability distribution over board states by performing depth-limited CFR to approximate an opponent's Nash equilibrium strategy. The CFR uses unsafe subgame solving (Ganzfried and Sandholm, 2015) over a sampled subset of possible board states, and it uses Stockfish to evaluate leaf nodes. For its own move, the bot maximizes its expected win probability.
Oracle	Exhaustively tracks board states. Chooses sense action to identify possible checks, or otherwise minimizes the expected number of possible board states. Chooses the move that is recommended by Stockfish most across all possible board states.
Gnash	Tracks all possible board states as well as all boards the opponent might believe are possible and probability estimates over each. These probabilities are calculated by assuming both players choose moves the same way. Gnash senses to minimize expected number of remaining states, and moves to maximize the Stockfish score of the state reached after a short playout. Tractable evaluation of scores is done by time-limited prioritized search, scoring likely-good moves on likely-true boards first.
Marmot	Tracks all possible opponent board states for the current time and past timesteps based on current observations. Uses a modified Monte Carlo counterfactual regret minimization (MC-CFR) algorithm for sensing and moving with a heuristic evaluation function (which includes Stockfish's board-evaluation function) based on determinized board position and a tracked uncertainty measurement to evaluate the intermediate states reached from action sequences sampled using MC-CFR. Employs a novel algorithm to take samples starting from a finite time horizon on the past, assuming the opponent knows the board state at that point, because the game tree is too large for complete MC-CFR starting from the beginning of the game.
DynamicEntropy and Frampt	Chooses sense and move actions by information set Monte Carlo tree search. The branching factor is reduced by screening dominated actions (e.g. sense options that provide strictly less information than others). Sampling of non-choice nodes is biased toward uniform to reflect opponent uncertainty. In DynamicEntropy, leaf nodes are evaluated by Stockfish. In Frampt, leaf nodes are evaluated by counting material and checks.
GarrisonNRL	Maintains a sample of up to 200 possible boards. Selects sensing action as the location that maximizes a sensing weight: the product of the number of potential opponent moves to a location and the value of pieces moving to those locations. For each board in the board sample, chooses a best move by attacking the opponent king if possible, otherwise by using the Leela chess engine with the Maia 1700 network (McIlroy-Young et al., 2020). Selects a movement action by using the most common best move.
trout (baseline)	Maintains a single board-state estimate that is formed directly from the latest observation of each square. Chooses the move recommended by Stockfish for its board estimate. If a piece was just captured or it thinks it will capture a piece next turn, it senses over the capture square. Otherwise it chooses a random location to sense that does not contain any of its own pieces.
attacker (baseline)	Randomly chooses and executes one of four scripted attack sequences. If that fails, no more moves are made.

Bot Name	**Brief Description**
URChIn (Unsupervised Representations for Chess Inference)	Employs a language model to obtain a "plausible state" distribution, which is sampled when deciding move actions. The language model is Word2Vec with hierarchical softmax (Mikolov et al., 2013), originally trained on ChessDB (Kirby, 2007) and updated after every RBC game. Move actions are selected by sampling the 20 highest-likelihood states according to the language model and determining the best worst-case move across these sampled states, according to StockFish. Senses to maximize the number distinct piece types across states. URChIn underperformed due to bugs, and has been improved following the tournament.
random (baseline)	Chooses moves uniformly at random.

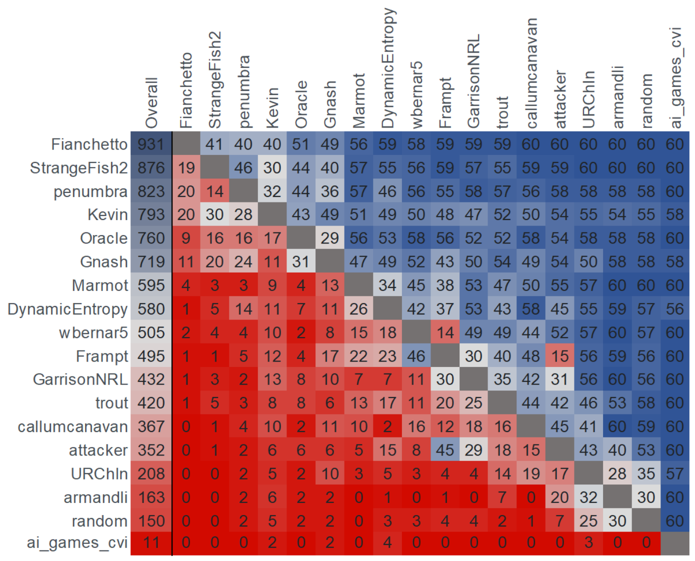
Table 2: High-level comparison of features included in competing algorithms.

Bot	Elo	Elo rank	Tracks all board states	Tracks opponent infosets	Tracks full game state	Models opponent moves	Models opponent senses	Uses a chess engine	Senses to minimize states
Fianchetto	1759	1	●			●		●	
StrangeFish2	1662	2	●					●	
penumbra	1584	3	●	●		●	●		
Kevin	1544	4	●	●		●	●	●	
Oracle	1503	5	●					●	●
Gnash	1454	6	●	●		●		●	●
Marmot	1315	7	●	●		●	●	●	
DynamicEntropy	1299	8	●					●	
Frampt	1208	10	●						
GarrisonNRL	1140	11				●		●	●
trout	1127	12						●	
callumcanavan	1066	13							
attacker	1049	14							
URChIn	854	15						●	●
random	753	17							

6. Results and Observations

After 30 rounds and 9,180 games, the bot Fianchetto won convincingly, with a 91% overall win rate and at least a 66% win rate against every opponent. The overall and per-pairing wins are displayed in Figure 1. Compared to the 2019 tournament, a greater fraction of this year's agents exceeded the baseline performance of the bot trout. Despite being a naïve application of a chess engine, trout managed at least one win against every opponent. By comparison to repeat participants, it can be estimated that the top four bots in this tournament matched or exceeded the performance of the inaugural tournament's winner.

Bayesian Elo ratings for all bots fluctuated in the early rounds, but the winning agent's dominance was apparent by the round 15 midpoint of the tournament. Figure 2 shows the estimated Elo and 95% confidence limits for each bot after each of the tournament's 30 rounds. The left axis shows ratings computed from all games, while the right axis shows



	Overall	Fianchetto	StrangeFish2	penumbra	Kevin	Oracle	Gnash	Marmot	DynamicEntropy	wbernar5	Frampt	GarrisonNRL	trout	callumcanavan	attacker	URChIn	armandli	random	ai_games_cvi
Fianchetto	931		41	40	40	51	49	56	59	58	59	59	59	60	60	60	60	60	60
StrangeFish2	876	19		46	30	44	40	57	55	56	59	57	55	59	59	60	60	60	60
penumbra	823	20	14		32	44	36	57	46	56	55	58	57	56	58	58	58	58	60
Kevin	793	20	30	28		43	49	51	49	50	48	47	52	50	54	55	54	55	58
Oracle	760	9	16	16	17		29	56	53	58	56	52	52	58	54	58	58	58	60
Gnash	719	11	20	24	11	31		47	49	52	43	50	54	49	54	50	58	58	58
Marmot	595	4	3	3	9	4	13		34	45	38	53	47	50	55	57	60	60	60
DynamicEntropy	580	1	5	14	11	7	11	26		42	37	53	43	58	45	55	59	57	56
wbernar5	505	2	4	4	10	2	8	15	18		14	49	49	44	52	57	60	57	60
Frampt	495	1	1	5	12	4	17	22	23	46		30	40	48	15	56	59	56	60
GarrisonNRL	432	1	3	2	13	8	10	7	7	11	30		35	42	31	56	60	56	60
trout	420	1	5	3	8	8	6	13	17	11	20	25		44	42	46	53	58	60
callumcanavan	367	0	1	4	10	2	11	10	2	16	12	18	16		45	41	60	59	60
attacker	352	0	1	2	6	6	6	5	15	8	45	29	18	15		43	40	53	60
URChIn	208	0	0	2	5	2	10	3	5	3	4	4	14	19	17		28	35	57
armandli	163	0	0	2	6	2	2	0	1	0	1	0	7	0	20	32		30	60
random	150	0	0	2	5	2	2	0	3	3	4	4	2	1	7	25	30		60
ai_games_cvi	11	0	0	0	2	0	2	0	4	0	0	0	0	0	0	3	0	0	

Figure 1: Crosstable of wins each bot had against each other bot in the competition (1020 total games for each bot, 60 per opponent).

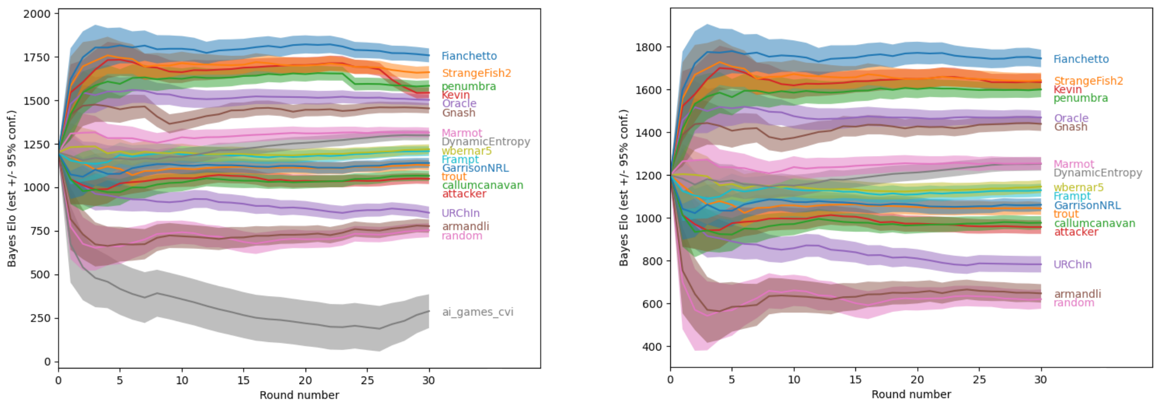


Figure 2: Convergence of Bayesian Elo ratings over the 30 round tournament. Elo estimates are shown with 95% confidence interval shaded. Left: total tournament rating. Right: algorithm strength rating, excluding forfeit games.

ratings calculated without forfeit results, which may be more useful in comparing algorithm strength. For example, the race for second place was extremely close between StrangeFish2, penumbra, and Kevin, although the latter two missed several games resulting in lower overall Elo.

Uncertainty Management: Most bots in this competition were based on exhaustive tracking of possible board states. It is evident from retracing tournament games that minimizing the number of possible board states was critical to overall performance. The

"Median # States" column in Table 3 shows the median number of possible board states from each agent's point of view *after sensing* on each turn of each game. The top 11 performers kept this median below 100, typically at or below 20. The lower performers had median numbers orders of magnitude larger. For this computation, any game replay which reached a number of states greater than one million was terminated, and the number was assumed to remain above one million for the rest of the game.

The minimization of possible board states does not determine algorithm strength, however. There is little difference in this quantity among top performers. The "States Δ Rank" column in Table 3 compares the ranked ordering of median states to overall tournament placement. Agents penumbra and StrangeFish2 used sense strategies with objectives other than minimization of possible board states, and exhibited overall tournament performance which exceeded their relative rank in median states, indicating they may have identified improvements in sensing strategies.

It is important to note that the median number of board states computed here is *not* the median number of RBC game states. The full game state also includes the history of sense and move choices, which together reflect all of what each player knows. The number of possible game states computed using the observation history is much larger than the possible arrangements of pieces on the board. No bot has made much progress in representing or using it. Improvement in this regard may require a non-exhaustive alternative to the hypothesis tracking which was common in this tournament.

Comparison to Chess Engine: Evaluation of moves and board states was frequently delegated to chess engines such as Stockfish (Romstad et al., 2018). Position strength remains important in RBC as it is in chess, and some agents augment chess engine values with RBC-specific heuristics, but in theory the use of chess evaluation in RBC limits solution strength. As discussed with regards to game state computation, the full RBC game state includes all the information available to each player, not just the arrangement of pieces on the board. A comprehensive evaluation of a game state then must also incorporate this information.

In general, though, move strength as assessed by a chess engine correlates well with performance in this tournament. This was computed by getting the top three moves from Stockfish given the true board state on each turn of each game. RBC-only conditions were handled by picking any move that captures the opponent's king if able, otherwise any move that gets out of check if necessary. The "Engine Move Agreement" column of Table 3 shows the percent of an agent's moves that were in the top three chess moves. Largely, better agreement with the engine corresponds with higher tournament placement. Two agents placed notably higher in the tournament compared to their move agreement rank: penumbra, which learned state values from reinforcement learning on RBC games rather than relying on an engine, and Marmot, which did use Stockfish evaluation, but in an uncertainty-aware framework adapted from MC-CFR.

Inspection of opening moves reveals some of the bias caused by chess engine evaluation in RBC. In this tournament, the most common opening moves mirrored those of chess—advance a pawn to e4 or d4, or a knight to f3—and did not reflect RBC-specific move strength. Games opened with e4 37% of the time, and of those games, white won 71%. However, the second most common opening, d4, was played 17% of the time and resulted in 66% losses for white. (The next most common opening move with worse performance was to pass without moving.) Players opened by advancing a pawn to c4 in only 5% of games, but were rewarded with 83% wins. As algorithms advance state evaluation functions specific to RBC rather than chess, it is likely that fewer games will open with d4, and more with c4.

Table 3: A comparison of each bot's tournament rank to their relative uncertainty management and use of traditional chess moves.

Bot	Elo	Elo rank	Median # States	# States Rank	# States Δ Rank	Engine Move Agreement	Move Agree. Rank	Move Agree. Δ Rank
Fianchetto	1759	1	9	2	+1	72%	2	+1
StrangeFish2	1662	2	15	5	+3	59%	4	+2
penumbra	1584	3	24	10	+7	43%	7	+4
Kevin	1544	4	13	3	-1	67%	3	-1
Oracle	1503	5	13	4	-1	73%	1	-4
Gnash	1454	6	18	7	+1	58%	5	-1
Marmot	1315	7	20	9	+2	28%	11	+4
DynamicEntropy	1299	8	16	6	-2	39%	9	+1
wbernar5	1219	9	5	1	-8	28%	12	+3
Frampt	1208	10	19	8	-2	25%	13	+3
GarrisonNRL	1140	11	61	11	0	44%	6	-5
trout	1127	12	3243	14	+2	36%	10	-2
callumcanavan	1066	13	7158	15	+2	8%	16	+3
attacker	1049	14	>1M	17	+3	4%	17	+3
URChIn	854	15	124	12	-3	39%	8	-7
armandli	777	16	204	13	-3	15%	14	-2
random	753	17	68263	16	-1	8%	15	-2
ai_games_cvi	288	18	-	-	-	-	-	-

7. Conclusions

The NeurIPS 2021 tournament of RBC provided an accessible platform for the development and comparison of algorithms for automated decision making under uncertainty. Algorithmic advancements were made compared to the 2019 tournament; the top four bots in this competition appear to match or exceed the previous winner's performance. However, significant research challenges remain between the current top performers and a practically-optimal solution to RBC. The tournament organizers hope to continue to hold an annual RBC competition as part of NeurIPS until the core research challenges are addressed.

References

Noam Brown and Tuomas Sandholm. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 2017. ISSN 0036-8075. doi: 10.1126/science.aao1733. URL http://science.sciencemag.org/content/early/2017/12/15/science.aao1733.

Noam Brown, Tuomas Sandholm, and Brandon Amos. Depth-limited solving for imperfect-information games. In *Neural Information Processing Systems (NeurIPS) Deep Reinforcement Learning Workshop (DRL)*, 2018.

Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. Deep counterfactual regret minimization. In *International Conference on Machine Learning (ICML)*, 2019.

Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining deep reinforcement learning and search for imperfect-information games. In *Neural Information Processing Systems (NeurIPS)*, 2020. URL https://arxiv.org/abs/2007.13544.

C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4 (1):1–43, March 2012. ISSN 1943-068X. doi: 10.1109/TCIAIG.2012.2186810.

Neil Burch, Michael Johanson, and Michael Bowling. Solving imperfect information games using decomposition. 2014. URL http://arxiv.org/abs/1303.4441.

Murray Campbell, A. Joseph Hoane Jr, and Feng-hsiung Hsu. Deep blue. *Artificial Intelligence*, 134(1-2):57–83, 2002.

Paolo Ciancarini and Gian Piero Favini. Monte Carlo tree search in Kriegspiel. *Artificial Intelligence*, 174(11):670–684, 2010.

Gregory Clark. Deep synoptic Monte-Carlo planning in Reconnaissance Blind Chess. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Neural Information Processing Systems (NeurIPS)*, 2021. URL https://openreview.net/forum?id=Joy2imuk604.

Rémi Coulom. Bayesian elo rating, February 2006. URL https://www.remi-coulom.fr/Bayesian-Elo/.

Sam Ganzfried and Tuomas Sandholm. Endgame solving in large imperfect-information games. *AAAI*, 2015.

Ryan W. Gardner, Corey Lowman, Casey Richardson, Ashley J. Llorens, Jared Markowitz, Nathan Drenkow, Andrew Newman, Gregory Clark, Gino Perrotta, Robert Perrotta, Timothy Highley, Vlad Shcherbina, William Bernadoni, Mark Jordan, and Asen Asenov. The first international competition in machine Reconnaissance Blind Chess. *Proceedings of Machine Learning Research (PMLR)*, 123, 2020. URL http://proceedings.mlr.press/v123/gardner20a/gardner20a.pdf.

Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. In *Neural Information Processing Systems (NeurIPS) Deep Reinforcement Learning Workshop (DRL)*, 2016.

David Kirby. Chess DB, September 2007. URL http://chessdb.sourceforge.net/.

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning (ECML)*, 2006.

Viliam Lisý, Marc Lanctot, and Michael Bowling. Online Monte Carlo counterfactual regret minimization for search in imperfect information games. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 27–36, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-3413-6. URL http://dl.acm.org/citation.cfm?id=2772879.2772887.

Reid McIlroy-Young, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Aligning superhuman ai with human behavior: Chess as a model system. *Proceedings of the 25th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2020. URL https://arxiv.org/abs/2006.01855.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv*, September 2013. URL https://arxiv.org/abs/1301.3781.

Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017. ISSN 0036-8075. doi: 10.1126/science.aam6960. URL http://science.sciencemag.org/content/356/6337/508.

Andrew J. Newman, Casey L. Richardson, Sean M. Kain, Paul G. Stankiewicz, Paul R. Guseman, Blake A. Schreurs, and Jeffrey A. Dunne. Reconnaissance blind multi-chess: An experimentation platform for ISR sensor fusion and resource management. In *Signal Processing, Sensor/Information Fusion, and Target Recognition XXV*, volume 9842, 2016.

Gian-Carlo Pascutto and Gary Linscott. Leela Chess Zero, January 2018. URL https://github.com/LeelaChessZero/lc0.

Gino Perrotta and Robert Perrotta. StrangeFish, October 2019. URL https://github.com/ginop/reconchess-strangefish.

Tord Romstad, Marco Costalba, and Joona Kiiski. Stockfish. https://stockfishchess.org/, 2018.

Stuart Russell and Jason Wolfe. Efficient belief-state AND-OR search, with application to Kriegspiel. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017a. URL http://arxiv.org/abs/1712.01815. http://arxiv.org/abs/1712.01815.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550: 354–359, 10 2017b. URL http://dx.doi.org/10.1038/nature24270.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018. URL https://science.sciencemag.org/content/362/6419/1140.

Michal Sustr, Vojtech Kovarík, and Viliam Lisý. Monte Carlo continual resolving for online strategy computation in imperfect information games. 2018. URL http://arxiv.org/abs/1812.07351. Available at http://arxiv.org/abs/1812.07351.

Gerald Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3), March 1995.

Brian Hu Zhang and Tuomas Sandholm. Subgame solving without common knowledge. In *Neural Information Processing Systems (NeurIPS)*, 2021. URL https://arxiv.org/pdf/2106.06068.pdf.

Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1729–1736. Curran Associates, Inc., 2008. URL http://papers.nips.cc/paper/3306-regret-minimization-in-games-with-incomplete-information.pdf.