The First International Competition in Machine Reconnaissance Blind Chess

Ryan W. Gardner ryan.gardner@jhuapl.edu
Corey Lowman corey.lowman@jhuapl.edu
Casey Richardson casey.richardson@jhuapl.edu
Ashley J. Llorens ashley.llorens@jhuapl.edu
Jared Markowitz jared.markowitz@jhuapl.edu
Nathan Drenkow nathan.drenkow@jhuapl.edu
Andrew Newman andrew.newman@jhuapl.edu
Johns Hopkins University Applied Physics Laboratory

Gregory Clark gregoryclark@google.com
Google

Gino Perrotta gino.mm.perrotta@gmail.com
The George Washington University

Robert Perrotta raaperrotta@gmail.com
Independent

Timothy Highley highley@lasalle.edu
La Salle University

Vlad Shcherbina vlad.shcherbina@gmail.com
Independent

William Bernadoni wbernar5@jhu.edu
Johns Hopkins University

Mark Jordan mark.jordan@src.tech
SRC Technologies

Asen Asenov bonumpro@gmail.com
XA, Inc.

Editors: Hugo Jair Escalante and Raia Hadsell

Abstract

Reconnaissance blind chess (RBC) is a chess variant in which a player cannot see her opponent's pieces but can learn about them through private, explicit sensing actions. The game presents numerous research challenges, and was the focus of a competition held in conjunction with of the 2019 Conference on Neural Information Processing Systems (NeurIPS). The 22 bots that played in the tournament leveraged a diverse set of algorithms, including variations of multi-state tracking, piece-wise probability estimation, Gibbs sampling, bandit algorithms, tree search, counterfactual regret minimization (CFR), deep learning, and others. None of the algorithms of which we are aware converges to an optimal strategy. Top algorithms generally incorporated sensing strategies that successfully minimized uncertainty (as measured in the number of possible opponent states). The top two approaches reduced this raw uncertainty metric less than some others. Successful strategies sometimes defied conventional wisdom in chess, as evidenced by deviations between win rate and aggregate move strength as assessed by the leading available chess engine.

Keywords: reconnaissance blind chess, imperfect information, reinforcement learning

1. Introduction

Games can provide useful environments for advancing machine learning research when they reflect essential challenges inherent in real-world applications. Even games with simple rules can require advanced techniques to play effectively, and small changes in game mechanics can yield distinctly different research challenges. Breakthroughs in classic games like checkers, chess, and Go depended on planning algorithms such as minimax and Monte Carlo tree search. Other games, such as poker, involved sophisticated approaches to decision making under uncertainty. The combination of these two aspects is rare, as few games require long term planning in the face of uncertainty.

For these reasons, researchers at the Johns Hopkins University Applied Physics Laboratory (JHU/APL) and other organizations proposed and hosted a competition as part of the Conference on Neural Information Processing Systems (NeurIPS) 2019 on reconnaissance blind chess (RBC) (Newman et al., 2016; Markowitz et al., 2018). RBC is like standard chess except a player cannot see where her opponent's pieces are a priori. Rather, she learns partial information about them through private sensing actions and the results of moves.

This paper, compiled by several of the competition's organizers and participants, summarizes the outcomes of the competition, describing background, selected approaches, and results. We offer several observations regarding performance, uncertainty management (measured by counting the number of possible opponent states), and aggregate move strength (as assessed by the leading available chess engine, Stockfish). Since the strongest algorithms used a variety of heuristics and no employed approach of which we are aware converges to an optimal solution, RBC remains a promising game for additional research.

2. The Game

RBC is a modification of chess. The essential differences are: A player cannot see where her opponent's pieces are *a priori*. Prior to each of her own moves, each player gets a sense action where she selects a 3×3 square of the chess board, unrevealed to the opponent, and is provided perfect information about the square (pieces and locations).[1] A player also learns when one of her pieces is captured or makes a capture, but she is not informed of the type of the corresponding opponent piece. The game has no notion of check. One wins by capturing the opponent's king or when the opponent runs out of time.

We refer the reader to `https://rbc.jhuapl.edu` for the complete list of rules and to get a hands-on feel by playing.

3. Research Challenges and Related Work

We outline research challenges of RBC briefly. A separate draft paper characterizes the game's complexity and challenges in more detail (Markowitz et al., 2018).

Approximating optimal actions in games of imperfect information is very different than doing so in games of perfect information. For example, in general, an optimal strategy in an imperfect information game is mixed (probabilistic) rather than pure (deterministic). In a

1. Many variants of RBC are possible including those with sensors of different sizes, limited sensing resources, and even noisy sensors.

game of imperfect information, one must consider the entire game tree (including states that are unreachable from the current state of the game) when approximating an optimal action. Approaches like Monte Carlo tree search (MCTS) (Browne et al., 2012), which was central to AlphaZero (Silver et al., 2017), for example, are not directly applicable. A variant of MCTS for situations with imperfect information, information set MCTS (ISMCTS) (Cowling et al., 2012) has no optimality guarantees.

State of the art approaches to approximating optimal solutions in imperfect information games like counterfactual regret minimization (CFR) (Zinkevich et al., 2008) are not practical for games as large as RBC.

For poker, researchers significantly reduced the size of the game by creating abstractions (Brown and Sandholm, 2017a; Sandholm, 2010). For example, one could consider all king-high flushes to be the same state and consider a bet of \$1,004 to be equivalent to a bet of \$1,000. However, no similar abstractions that would not significantly reduce the fidelity of the original game are are obvious for RBC.

Techniques have also been developed that enable sound decomposition of imperfect information games into subgames (Burch et al., 2014; Brown and Sandholm, 2017b; Sustr et al., 2018), which can drastically reduce the size of the game about which one must reason. These were also central to famous superhuman poker systems (Brown and Sandholm, 2017a; Moravčík et al., 2017). However, these techniques operate on the *public* game tree, i.e., where each state is indistinguishable given knowledge shared by all players. In RBC, a player does not know where her opponent has sensed and thus has very little knowledge about what her opponent knows. As a result, there is little notion of a *public* game and a useful means of decomposing RBC is not obvious.

Other games share some properties of RBC. Kriegspiel is a blind chess variant (Favini, 2010; Ciancarini and Favini, 2007, 2010; Russell and Wolfe, 2005) where a player mostly learns about her opponent's positions through umpire feedback on move legality, captures, and check. The ability to sense portions of the board in RBC gives players more ability to manage uncertainty, which may make the game more realistic for many scenarios. Phantom Go (Cazenave, 2006; Wang et al., 2018) is the Go equivalent of Kriegspiel for chess.

Dark chess is another blind chess variant where a player can see the squares to which they can legally move. It may be among the most similar games to RBC, from a feel and research-challenge perspective. Like Kriegspiel, it omits explicit sensing, potentially resulting in less ability to manage one's uncertainty and less uncertainty about what one's opponent knows, two key elements of RBC.

Banqi uses a different board and pieces than chess although it is sometimes called dark chess, half chess, or blind chess. Banqi can be modeled as a game of perfect information with a chance element since no player knows more than another at any point in the game.

4. Competition Structure

The NeurIPS RBC competition was open to anyone in the world who wanted to build a bot.[2] The competition consisted of a tournament with 12 rounds. Each round was a round-robin where each bot played every other bot twice; once as white and once as black. In

2. Some potential competitors, such as those under the age of 18 or employees of Johns Hopkins University, were not legally eligible for the cash prize.

total, there were 5,544 games. Ranking was determined simply by the number of wins. 22 bots participated in total; 15 competitors and 7 baseline bots from JHU/APL.

Each player had a 15 minute time limit to make all their moves in a given game. The organizers offered a (mostly symbolic) 1,000 USD cash prize to the first place winner.

5. Overview of Approaches

The algorithms used for RBC bots were diverse. We briefly outline the core approaches for several select bots in Table 1. Although a few of these bots did not perform well due to technical reasons, we include their descriptions because the approaches may prove valuable in the future. Additionally, their performance for this competition illustrates some of the practical challenges of fully executing such approaches in a relatively short time frame.

Table 1: Overview of algorithms used by select bots.

[Rank] Bot	**Brief Description**
[18] penumbra	*Underperformed in the tournament due to hardware issues.* Tracks all possible opponent board states. Samples opponent states with Gibbs sampling and chooses move and sense actions based on fixed-length playouts. Explores actions according to an upper confidence bound (UCB) bandit algorithm. Evaluates states and chooses playout actions with a ten block residual network which was trained on historical RBC games.
[15] random (baseline)	Chooses moves and senses uniformly at random.
[12] trout (baseline)	Maintains a single board-state estimate that is formed directly from the last 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.
[11] Bonum	*Underperformed in the tournament due to connection problems and Stockfish crashes.* Uses a rule-based algorithm to maintain a list of possible positions for each opponent piece, inferring unknown probabilities from opponent-move-strength according to Stockfish. Uses a set of heuristics to choose sensing locations, attempting to minimize spread of probable piece positions and gain strategic knowledge. Selects the move recommended by Stockfish-combined-with-heuristics for the most probable piece configuration.
[6] Marmot (baseline)	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, leveraging online outcome sampling (Lisý et al., 2015) to importance sample histories that are consistent with the current history of the game. It uses a heuristic 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 as a surrogate for a rollout algorithm. Employs a novel algorithm to take samples starting from a finite time horizon on the past, because the game tree is too large for complete MC-CFR.
[5] wbernar5	Tracks up to 500 potential states chosen uniformly by advancing the prior turn's tracked states. Senses to maximize the number of states eliminated in the worst case. Chooses the move with the best worst-Stockfish score across all tracked boards. If all tracked states are eliminated as impossible, creates a new set of potential states by re-simulating from a checkpoint.

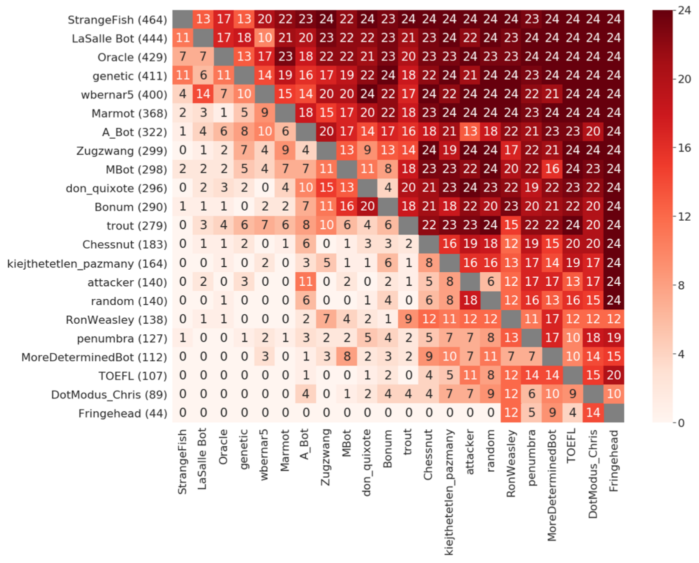
[Rank] Bot	Brief Description
[4] genetic	Tracks all possible opponent board states. Assumes the opponent can choose any observation-consistent placement of its pieces at each decision point. Builds the extensive form game tree from the current possible states up to four levels (i.e. opponent chooses initial placement, bot move, opponent sense, opponent move). (Usually 3 are used for performance.) Evaluates leaves using a custom chess engine, plus a term for each player's uncertainty (how many board states are possible from the player's viewpoint). Approximately solves the partial game tree using CFR (Zinkevich et al., 2008). Chooses actions using CFR results with the addition of heuristics. (E.g., moves that put the opponent in check are artificially encouraged.)
[3] Oracle (baseline)	Tracks all possible opponent board states. Nominally chooses the sensing action that minimizes the expected number of possible board states assuming each has equal probability, with some heuristics to choose an alternative check-detecting sense if Oracle may be in check. Chooses the move that is recommended by Stockfish most across all the possible board states.
[2] LaSalle Bot	Maintains 32 probability distributions, one for the location of each piece. Senses the square with maximum "uncertainty score", which is the square where a piece has the probability of being in that square that is closest to 50%, but then adjusted by multiple heuristics (e.g. ignore pieces that cannot attack, potential check is more important than other squares, pawns are less important than the queen, etc.). Uses Stockfish to evaluate moves on multiple generated boards and averages the results to choose a move. Described in more detail in a separate paper (Highley et al., 2020).
[1] StrangeFish	Tracks all possible board states. Evaluates players' relative advantage on any board state using Stockfish plus custom heuristics for RBC-unique game states. Uses the calculated advantage scores (1) to estimate the probability of a hypothetical board being the true state, (2) as a proxy for game rollouts when considering move choices, and (3) to assess sensing importance by score disparity in move choices across the remaining hypothetical board states. Chooses moves to maximize the change in position advantage averaged across a random sample of possible board states. Chooses sense locations to maximize expected influence on the following move decision.

6. Results and Observations

The competition results are summarized by the cross table in Figure 1. StrangeFish was a clear winner with 20 more total wins than the second-place bot and a winning record against every bot individually.

A Naive Separator: We observe two tiers of bot performance, with the baseline bot trout separating the two tiers. Trout constitutes what might be considered a natural, first-cut *naive* approach to a bot. It maintains a single, simple estimate of the board directly derived from the most recent observation of each square and using moves recommended by the strongest freely-available chess engine, Stockfish. The fact that trout (whose source code is available with our development kit and is displayed on our documentation website) formed a rough separator with many bots performing below it illustrates the difficulty of improving upon naive approaches.

Managing Uncertainty: To provide some insight into the relationship between uncertainty and bot success, we examined all the games from the NeurIPS competition and computed the number of opponent states that were possible from each bot's perspective prior to making each move (i.e., after receiving the results of each sense action). This provides one intuitive measure of uncertainty, but is not a complete measure. We imposed a maximum of 50,000 possible states and stopped tracking once the number of states exceeded



	StrangeFish	LaSalle Bot	Oracle	genetic	wbernar5	Marmot	A_Bot	Zugzwang	MBot	don_quixote	Bonum	trout	Chessnut	kiejthetetlen_pazmany	attacker	random	RonWeasley	penumbra	MoreDeterminedBot	TOEFL	DotModus_Chris	Fringehead
StrangeFish (464)		13	17	13	20	22	23	24	22	24	23	24	24	24	24	24	24	23	24	24	24	24
LaSalle Bot (444)	11		17	18	10	21	20	23	22	22	23	21	23	24	22	24	23	24	24	24	24	24
Oracle (429)	7	7		13	17	23	18	22	22	21	23	20	23	23	24	23	23	24	24	24	24	24
genetic (411)	11	6	11		14	19	16	17	19	22	24	18	22	24	21	24	24	23	24	24	24	24
wbernar5 (400)	4	14	7	10		15	14	20	20	24	22	17	24	22	24	24	24	22	21	24	24	24
Marmot (368)	2	3	1	5	9		18	15	17	20	22	18	23	24	24	24	24	23	24	24	24	24
A_Bot (322)	1	4	6	8	10	6		20	17	14	17	16	18	21	13	18	22	21	23	23	20	24
Zugzwang (299)	0	1	2	7	4	9	4		13	9	13	14	24	19	24	24	17	22	21	24	24	24
MBot (298)	2	2	2	5	4	7	7	11		11	8	18	23	23	22	24	20	22	16	24	23	24
don_quixote (296)	0	2	3	2	0	4	10	15	13		4	20	21	23	24	23	22	19	22	23	22	24
Bonum (290)	1	1	1	0	2	2	7	11	16	20		18	21	18	22	20	23	20	21	22	20	24
trout (279)	0	3	4	6	7	6	8	10	6	4	6		22	23	23	24	15	22	22	24	20	24
Chessnut (183)	0	1	1	2	0	1	6	0	1	3	3	2		16	19	18	12	19	15	20	20	24
kiejthetetlen_pazmany (164)	0	0	1	0	2	0	3	5	1	1	6	1	8		16	16	13	17	14	19	17	24
attacker (140)	0	2	0	3	0	0	11	0	2	0	2	1	5	8		6	12	17	17	13	17	24
random (140)	0	0	1	0	0	0	6	0	0	1	4	0	6	8	18		12	16	13	16	15	24
RonWeasley (138)	0	1	1	0	0	0	2	7	4	2	1	9	12	11	12	12		11	17	12	12	12
penumbra (127)	1	0	0	1	2	1	3	2	2	5	4	2	5	7	7	8	13		17	10	18	19
MoreDeterminedBot (112)	0	0	0	0	3	0	1	3	8	2	3	2	9	10	7	11	7	7		10	14	15
TOEFL (107)	0	0	0	0	0	0	1	0	0	1	2	0	4	5	11	8	12	14	14		15	20
DotModus_Chris (89)	0	0	0	0	0	0	4	0	1	2	4	4	4	7	7	9	12	6	10	9		10
Fringehead (44)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	5	9	4	14	

Figure 1: Crosstable of wins each bot had against each other bot in the competition (24 total games per bot pair).

50,000 because the computation starts to become intractable at that point. After the number of states exceeded 50,000, we assumed it remained above 50,000 for the remainder of the game. We computed the median number of possible opponent states across all turns along with a bot rank in terms of this median value and present the results in Table 2.

Table 2: The median number of possible opponent states from each bot's perspective prior to making a move through the NeurIPS competition, along with a ranking of that median compared to the bot's ranking in the tournament.

Bot	**Median # States**	**# States Rank**	**Competition Rank**	**Δ Rank**
A_Bot	12	1	7	+6
Oracle (baseline)	13	2	3	+1
MoreDeterminedBot (baseline)	13	3	19	+16
wbernar5	18	4	5	+1
LaSalle Bot	19	5	2	-3
StrangeFish	19	6	1	-5
Marmot (baseline)	24	7	6	-1
RonWeasley	25	8	17	+9
penumbra	48	9	18	+9
genetic	49	10	4	-6
Bonum	81	11	11	0
don_quixote	358	12	10	-2
MBot	574	13	9	-4
Zugzwang (baseline)	14,435	14	8	-6
Chessnut	50k+	15	13	-2
trout (baseline)	50k+	16	12	-4
TOEFL	50k+	17	20	+3

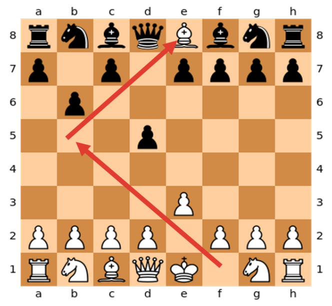


Figure 2: The shortest games in the NeurIPS competition.

Bot	**Median # States**	**# States Rank**	**Competition Rank**	**Δ Rank**
DotModus_Chris	50k+	18	21	+3
attacker (baseline)	50k+	19	16	-3
kiejthetetlen_pazmany	50k+	20	14	-6
random (baseline)	50k+	21	15	-6

One can see that bot rank tends to improve with the ability to reduce the number of possible opponent states. At the same time, these data also indicate that one's sensing strategy must go beyond minimizing the number of possible opponent states. StrangeFish, for example, was ranked number 6 by this metric. It had a median number of states 58% higher than A_bot's and 46% higher than Oracle's but outperformed both of them.

Move Strength in Chess: Chess engines like Stockfish provide a quick way to evaluate move selection. At the same time, "classic" chess moves are not necessarily good RBC moves. Figure 2 illustrates an example of the shortest type of game that occurred in the competition. It also illustrates the discrepancy between move strength in classic chess and in RBC. Advancing the queen pawn, as done by black, is typically considered a reasonable early move in classic chess. Advancing the bishop, as done by white, is considered a weak move in classic chess because of the simple defense by black.

We examined move strength in competition games as measured by Stockfish compared to competition success. For the 5,544 games of the tournament, we queried Stockfish with the full state of the player's pieces prior to each move. We identified how each bot's moves ranked in Stockfish's recommendations. We refer to that rank of each move as the *move score*. Table 3 provides the mean move score for each bot, and, like Table 2, ranks each player according to that mean score and also provides the difference in that rank compared to the bot's rank in the competition.

Table 3: The mean move score of each bot's moves through the NeurIPS competition as the moves ranked according to the Stockfish chess engine, along with a ranking of that mean compared to the bot's ranking in the tournament.

Bot	**Mean Move Score**	**Move Rank**	**Competition Rank**	**Δ Rank**
LaSalle Bot	4.47	1	2	+1

Bot	Mean Move Score	Move Rank	Competition Rank	Δ Rank
Oracle (baseline)	4.59	2	3	+1
StrangeFish	4.67	3	1	-2
A_Bot	4.82	4	7	+3
don_quixote	6.23	5	10	+5
wbernar5	6.25	6	5	-1
MBot	6.34	7	9	+2
trout (baseline)	6.35	8	12	-4
RonWeasley	7.43	9	17	+8
Bonum	7.68	10	11	+1
genetic	7.74	11	4	-7
Zugzwang (baseline)	8.15	12	8	-4
kiejthetetlen_pazmany	8.36	13	14	+1
Marmot (baseline)	9.82	14	6	-8
DotModus_Chris	11.58	15	21	+6
penumbra	11.97	16	18	+2
MoreDeterminedBot (baseline)	12.56	17	19	+2
attacker (baseline)	13.85	18	16	-2
random (baseline)	14.98	19	15	-4
Chessnut	15.15	20	13	-7
TOEFL	16.29	21	20	-1

We see a strong correlation between move rank and competition rank. However, there are significant outliers. The most notable outliers who outperformed their competition rank in the tournament are Marmot, genetic, and Chessnut. Marmot and genetic used modified counterfactual regret minimization approaches, which are intended to approximate optimal probabilities of actions accounting for the opponent's probabilistic knowledge. These outliers suggest that these bots' reasoning over probability may have provided some strength that was lost by a less-thorough analysis of potential long-term move impacts.

Nature of the Approaches: Another observation is that the strongest performing bots from this competition used effective heuristics to minimize some measurement of uncertainty. They also heavily leveraged chess engines, which assumes a determinization of the game. Determinization is known the have significant theoretic problems (Koller and Pfeffer, 1994). The fact that the strongest algorithms were based on these heuristics with known problems indicates that much research lies ahead with respect to RBC.

7. Conclusions

RBC provides an accessible testbed for AI research that captures two essential aspects of real-world problems - uncertainty and sensing. The RBC NeurIPS competition provided new insights into the strengths and limitations of many popular AI techniques in addressing challenges in decision-making under uncertainty.

The many open research challenges encourage us to continue to promote research in RBC through various forums.[3] We also hope to facilitate study of variants of the game that could draw focus to different decision making aspects and algorithmic strengths. These

3. At the time of writing, we have a dynamic online leaderboard with a cash prize offered to the top-ranked bots at the end of August 2020.

include variants where the pieces do not start in a known state, where there is noise in the received information (Newman et al., 2016), and where a player is notified if an opponent intentionally or unintentionally makes a pass move. It is our hope that continued research into RBC and other imperfect information games will continue to advance the state of the art in this critical research area.[4]

References

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

Noam Brown and Tuomas Sandholm. Safe and nested subgame solving for imperfect-information games. In *Neural Information Processing Systems (NIPS)*, 2017b.

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.

Tristan Cazenave. A phantom-go program. In *Advances in Computer Games*, 2006.

Paolo Ciancarini and Gian Piero Favini. Representing kriegspiel states with metapositions. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

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

P. I. Cowling, E. J. Powley, and D. Whitehouse. Information set Monte Carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):120–143, June 2012. ISSN 1943-068X. doi: 10.1109/TCIAIG.2012.2200894.

Gian Piero Favini. The dark side of the board: Advances in chess kriegspiel. Technical Report UBLCS-2010-06, University of Bologna, Department of Computer Science, 2010.

Timothy Highley, Brendan Funk, and Laureen Okin. Dealing with uncertainty: a PiecewiseGrid agent for reconnaissance blind chess. *The Journal of Computing Sciences in Colleges*, 35(8), 2020.

Daphne Koller and Avi Pfeffer. Generating and solving imperfect information games. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1994.

4. Game resources are available at https://rbc.jhuapl.edu.

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.

Jared Markowitz, Ryan W. Gardner, and Ashley J. Llorens. On the complexity of reconnaissance blind chess. 2018. URL http://arxiv.org/abs/1811.03119. Available at http://arxiv.org/abs/1811.03119.

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.

Stuart Russell and Jason Wolfe. Efficient Belief-State AND-OR Search, with Application to Kriegspiel. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

Tuomas Sandholm. The state of solving large incomplete-information games, and application to poker. *AI Magazine*, 31(4), 2010.

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. *CoRR*, abs/1712.01815, 2017. URL http://arxiv.org/abs/1712.01815.

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.

J. Wang, T. Zhu, H. Li, C. Hsueh, and I. . Wu. Belief-state monte carlo tree search for phantom go. *IEEE Transactions on Games*, 10(2):139–154, 2018.

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.