Hindawi
Complexity
Volume 2023, Article ID 9879785, 1 page
https://doi.org/10.1155/2023/9879785

Retraction

Retracted: Heuristic Sensing: An Uncertainty Exploration Method in Imperfect Information Games

Complexity

Received 19 December 2023; Accepted 19 December 2023; Published 20 December 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Manipulated or compromised peer review

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regret that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] Z. Guo, X. Wang, S. Qi, T. Qian, and J. Zhang, "Heuristic Sensing: An Uncertainty Exploration Method in Imperfect Information Games," *Complexity*, vol. 2020, Article ID 8815770, 9 pages, 2020.

Hindawi
Complexity
Volume 2020, Article ID 8815770, 9 pages
https://doi.org/10.1155/2020/8815770

Research Article

Heuristic Sensing: An Uncertainty Exploration Method in Imperfect Information Games

Zhenyang Guo,[1] **Xuan Wang**,[1] **Shuhan Qi**,[1,2] **Tao Qian**,[1] **and Jiajia Zhang**[1,2]

[1]*Harbin University of Technology Shenzhen, Shenzhen 518055, China*
[2]*Pingan-Hitsz Intelligence Finance Research Center, Shenzhen 518055, China*

Correspondence should be addressed to Jiajia Zhang; zhangjiajia@hit.edu.cn

Received 26 August 2020; Revised 6 October 2020; Accepted 9 October 2020; Published 24 October 2020

Academic Editor: Zhihan Lv

Imperfect information games have served as benchmarks and milestones in fields of artificial intelligence (AI) and game theory for decades. Sensing and exploiting information to effectively describe the game environment is of critical importance for game solving, besides computing or approximating an optimal strategy. Reconnaissance blind chess (RBC), a new variant of chess, is a quintessential game of imperfect information where the player's actions are definitely unobserved by the opponent. This characteristic of RBC exponentially expands the scale of the information set and extremely invokes uncertainty of the game environment. In this paper, we introduce a novel sense method, Heuristic Search of Uncertainty Control (HSUC), to significantly reduce the uncertainty of real-time information set. The key idea of HSUC is to consider the whole uncertainty of the environment rather than predicting the opponents' strategy. Furthermore, we realize a practical framework for RBC game that incorporates our HSUC method with Monte Carlo Tree Search (MCTS). In the experiments, HSUC has shown better effectiveness and robustness than comparison opponents in information sensing. It is worth mentioning that our RBC game agent has won the first place in terms of uncertainty management in NeurIPS 2019 RBC tournament.

1. Introduction

Game theory is the mathematical study of interaction among independent, self-interested players, providing a very simple but powerful paradigm to capture decision problem. The classical category divides games into perfect information games (PIGs) and imperfect information games (IIGs). In PIGs, players can obtain complete information of game environment. However, pervasively existing in real world, players cannot sense complete or reliable information of games. IIGs address these cases and model strategic interactions among agents with only partial or unreliable information. Thus, the exploitation of imperfect information of IIGs is one of the most critical challenges for game solving since how well players understand the game environment greatly influences the effectiveness of their strategies.

In this paper, we focus on a recently introduced IIG, reconnaissance blind chess for research of imperfect information exploitation. Reconnaissance blind chess is actually a family of games, and we only focus on one variant, which we will refer to as RBC for simplicity [1]. RBC was designed intentionally to add a certain amount of uncertainty by adjusting some rules of chess and adding an explicit sense step.

Furthermore, we utilize the algorithm of MCTS in this paper. Monte Carlo (MC) method has been used extensively in PIGs [2] and IIGs [3] which uses random simulations to approximate the true value of states in IIGs. Furthermore, Upper Confidence Bound for Trees (UCT) is the most popular MCTS algorithm, using upper confidence bounds, a formula trying to settle the exploitation-exploration dilemma, as a tree policy for selection and expansion [4]. UCT converges to Minimax, the optimal algorithm used for two-player zero-sum games [5], given enough time and memory. However, the reality is that time and memory are limited. Hence, one severe challenge of the MCTS + UCT structure is the contradiction between the accuracy of states' estimation and limited simulation time both of which are critical for a competitive game program.

The contribution of this paper is twofold. First, we introduce a novel sense method, HSUC, to effectively exploit and manage the uncertainty of the game environment. Our method is no longer entirely dependent on the accuracy of opponents' actions prediction which severely relies on plenty of simulation time. Instead of that, the key idea of HSUC is focusing on reducing the whole uncertainty of the environment, which is characterized by real-time information set in the game solving process. Second, we realize a practical framework for RBC game that incorporates HSUC with MCTS + UCT. NeurIPS 2019 tournament contains a final win rate rank and several ranks of different indicators. Our agent constructed with this framework has ranked the 7th in the final rank and won the first rank in terms of uncertainty management in particular.

2. Environment and Preliminaries

In this part, we will briefly introduce the rules of RBC, explain why RBC is a problem worth studying, and point out the difficulty of research. And then, we provide some preliminaries for later discussion.

2.1. Environment: RBC and Its Challenges. The major difference between RBC and standard chess is that RBC players are not informed of the opponent's actions in the process of the game. For managing this hidden information, an additional step called "sense" is embedded prior to the move step. During the sense step, a player selects a square of the chessboard and learns all pieces and their types within the square, and that action is invisible to another player. This step is the most important way for the player to obtain real information about the opponent. That means players should consider their sense strategies to choose a region to review an unknown part of the board. In addition, some changes have been made to other rules, for example, the player wins by capturing the opponent's king, but, in chess, a win occurs when the King is in under attack or in "check" and every possible move by the King will also put it in check. Since the player cannot see the opponent's chess pieces, some invalid actions may occur when the player moves. For details, please check the description of the website (https://rbc.jhuapl.edu/gameRules). The game tree of RBC is shown in Figure 1.

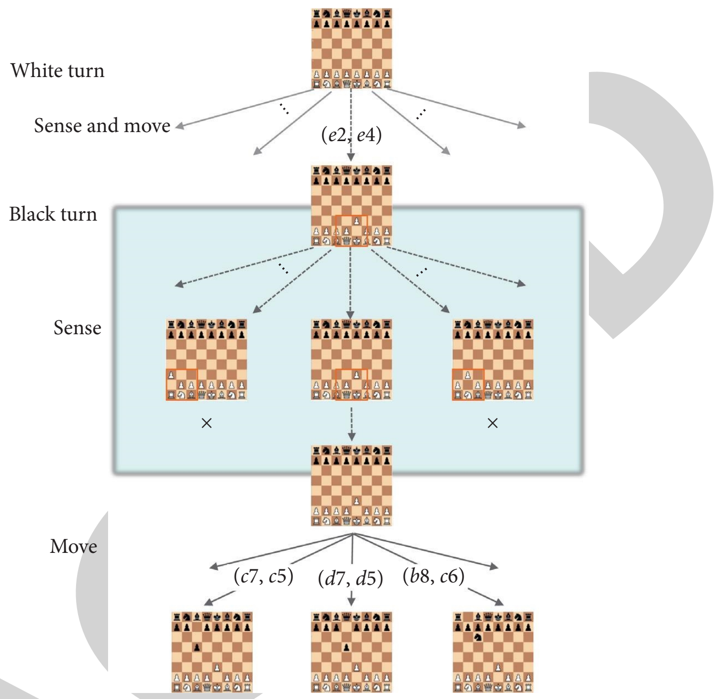

Figure 1: Game tree of RBC. One turn in RBC containing two phases cannot be observed by the opponent: sense and move. In the sense phase, a sense square (area in the red box) is chosen based on sense strategy firstly. Then, positions in the sense square of the chessboard are checked and conflicting states are removed from the information set based on sense result. After sense, a move is selected based on the move strategy. Move action (c7, c5) means that moving the piece of pawn from positions c7 to c5 and (b8, c6) means moving the knight in b8 to c6.

The past decades have witnessed rapid progress in the ability of AI systems to play increasingly complex games, such as go of PIGs [6] and poker of IIGs [7]. Not long ago, Brown et al. proposed poker agent Pluribus to solve the problem of multiplayer poker [8]. But RBC, as an IIG, is even more complex in certain aspects than multiplayer poker. We will discuss challenges in RBC in the following two aspects: the game size and the number of possible states in the information set.

Generally speaking, the game size can be measured by the number of states that players may encounter in the game. A practical method to measure the game size proposed by Shannon in 1950 [9] is widely adopted. According to the method, the game size of Lim 2-P Poker (Lim 2-P Poker refers to Limit Heads-Up Texas Hold'em) is 10^{13}, the game size of chess is 10^{43}, and that of RBC is 10^{139} [10]. Table 1 lists the number of states for several representative games and it denotes that RBC can approximately achieve a level similar to No-Limit Poker and go in terms of game size.

IIGs' complexity can be measured by another metric: the average number of possible states in the information sets. In RBC, this metric represents how difficult it is to evaluate a given perceived state. Poor sense strategy may lead to an exponential growth of the scale of information sets. Thus, the key property of RBC is the information asymmetry, that is, the uncertainty about the opponent's information. Table 2 shows that Jared Markowitz et al. [10] have calculated that the approximate average number of states of real-time information set in RBC is 1.3×10^{68}, which is even larger than Six-Player-No-Limit Poker.

2.2. Preliminaries

2.2.1. Extensive-Form Games. Sequential games are normally formalized as extensive-form games in which one or more agents or players perform sequential interactions. The extensive-form game can be described as a conceptual mode of six-tuple $\langle P, H, Z, A_m(h), \sigma_c(h, a), u_p(z)\rangle$:

(1) P: the set of players.

Table 1: Approximate size of games.

Game	Size
Lim 2-P Poker	10^{13}
Chess	10^{43}
RBC	10^{139}
No-Lim Poker	10^{162}
Go (19×19)	10^{170}

Table 2: Approximate means the number of possible opponents' states in an information set.

Game	Game states
Heads-Up No-Limit	1083
Chess	1
RBC	1.3×10^{68}
Six-Player-No-Limit	6.4×10^{14}
Go (19×19)	1

(2) H: a finite set H of sequences, the possible histories of actions, such that the empty sequence is in H and every prefix of a sequence in H is also in H.

(3) Z: the set of all terminal states, corresponding to all leaf nodes in the game tree.

(4) $A(h)$: the set of legal actions from state $h(h \notin z)$, corresponding to all edges starting from node h in the game tree.

(5) $\sigma_c(h, a)$: the probability that chance will take action $a \notin A(h)$ from state h.

(6) $u_p(z)$: the payoff for player p if the game ends in state $z(z \in Z)$.

We can further define the notations in RBC based on the mode as follows.

The behavior of players in RBC is similar to that of chess, except that an additional sense step is added before the move step. Thus, each player's strategy in one turn contains two phases, sense and move.

In each turn, player i chooses actions (a sense action and a move action), by its strategy $\sigma^i = (\sigma^i_s, \sigma^i_m)$, σ^i_s is player i's sense strategy, and σ^i_m is move strategy.

Furthermore, action set $A(h)$ in RBC is generated from the set of legal sense actions $A_s(h)$ and the set of legal move actions $A_m(h)$, $A(h) = A_m(h) \times A_s(h)$. Specifically, a sense action $a_s = s \in A_s(h)$ locates a $3 * 3$ area centered on s to be sensed.

2.2.2. Information Set. For IIGs, information set I_p for each player p is a partition of H_p. For any information set $I \in I_p$, any two states $h, j \in I$ are indistinguishable to player p. Figure 2 uses RBC to give an example of information set.

In a game tree, I_p is a set of decision nodes of player p, which meets the following two conditions:

(1) Each decision node in I_p is the decision node of player p

(2) When the game reaches a decision node in I_p, player p knows that it is in I_p but does not know which decision node of I_p it encounters

3. Heuristic Search of Uncertainty Control

A heuristic method is an approach for problem solving or self-discovery, which is not guaranteed to be optimal, perfect, or rational, but sufficient for reaching a feasible solution. While finding an optimal solution is impossible or impractical, heuristic methods can be used to speed up the process of finding a satisfactory solution. Heuristic search refers to a search strategy that attempts to optimize a problem by iteratively improving the solution based on a given heuristic function or a cost measure [11].

The heart of heuristic search methods is the idea of "continual researching" where a sound local search procedure is invoked whenever the agent must act without retaining any memory about how or why to reach the current state [12]. Our method for RBC game can be seen as a kind of heuristic search methods, using some measure function for better information exploitation. For solving RBC game, we divide the problem into two subproblems: how to control the explosive growth in scale of information sets and how to choose the most beneficial move action under imperfect information during the game. These two parts make up our heuristic search strategy.

3.1. Heuristic Search for Sense Strategy in RBC Game. The characteristic of RBC brings several difficulties to heuristic search in game solving. Firstly, since we cannot definitely know the opponent's knowledge and strategy, unreliable information may lead to an incorrect search direction. Secondly, how to control the growth of game states in information sets is another challenge. In order to ensure that the subsequent search is performed correctly, we must retain all possible states of the opponent as the current information set and cannot casually abandon any state. In addition, the player has a variety of action options. These two aspects have led to an information set of space rapid explosion and brought difficulty to storage.

Considering the above difficulties, we propose a novel sense method, HSUC, to prevent scale of real-time information set from exponential growth. Although RBC rules provide some extra information, such as notification of sense results, move results, and whether the player captures pieces, which can be employed to help reduce the uncertainty, HSUC is the major method to exploit all hints of RBC in our game system.

In this section, we will discuss how HSUC works to minimize the scale of the information set in RBC. An ideal way is to predict the opponent's next move action (s_f, s_t) and then take sense action a_{s_t} to get the sense result. A practical method for predicting opponents' actions is to absorb the idea of self-play, which is to use our own strategy to simulate the opponents' actions. Whenever we act sense action, we obtain the opponent's most likely action for every

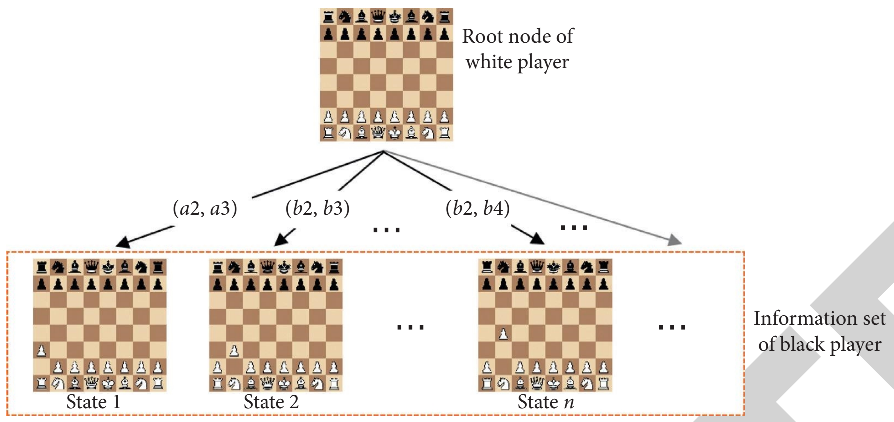

Figure 2: An example of the information set in RBC game. The root node is a white turn node. After the white player moves, the black player faces many indistinguishable possible states that are created by the opponent's unknown move; e.g., (a2, a3) and (b2, b3, are all possible move actions. These indistinguishable states in the dotted box form an information set of the black player.

remaining situation after the previous turn by move action selection strategy.

Unfortunately, the above approach is prone to bias. First of all, the initial premise of self-prediction is that the opponents adopt similar strategies to ours. When dealing with some specific agents, such as random or more powerful ones, sense actions will be severely misled which causes crashed performance of the whole system. Moreover, since the information set contains more than one state, plenty of sampling is required during the game tree search in order to guarantee the accuracy of state value evaluation.

HSUC focuses on estimating and reducing the whole uncertainty of the environment. Specifically in RBC, HSUC tries to find out the best sense square to minimize the number of possible states in real-time information set. Considering the game tree given in Figure 1, after the white player moves, the black player expands the game tree to form its real-time information set, which is the foundation of the next phase's strategy. To describe sense actions, the information set in the k-th turn is described as I_k: $I_k = \{h_0, h_1, \ldots, h_n\}$, $|I_k| = n$ is the number of possible states in I_k. Each sense action reveals a $3 * 3$ sense square and helps eliminate some impossible states from I_k. For example, if the sense action reveals no piece in the sense square, all states with pieces in the sense square can be determined as "impossible states" and removed from I_k. Let $A_s(I_k) = \{a_0, a_1, \ldots, a_t\}$ and $I_k(a_i)$ denote the reduced information set by taking sense action a_i. Then, $I_k(a_i) \subseteq I_k$, $m_i = |I_k(a_i)|$. Employing $g_k = ((|I_k| - |I_{k+1}|)/|I_k|) = 1 - (m_i/n)$ as the representative decay radio of sense action a_i, the target of heuristic search can be formalized as

$$a_i = \arg\max_{a_i \in A_s(I_k)} g_k = \arg\max_{a_i \in A_s(I_k)} \left(1 - \frac{m_i}{n}\right). \tag{1}$$

Here, g_k is employed to trace the tendency of game uncertainty. In this sense, the goal is to choose a sense action to maximize g_k for each turn of the game. However, there is no guarantee that the exact value of g_k can be found under the condition of imperfect information. So what should be done is to design a proper heuristic function H to evaluate g_k.

Firstly, we introduce how to evaluate the sense action's efficiency. Let $D_i(h_1, h_2)$ indicate whether sense action a_i can distinguish between states h_1 and h_2 in I_k. As long as one of the 9 positions in the $3 * 3$ sense square is different (existence or types of pieces), D_i is set to 1; otherwise, it is set to 0. Figure 3 shows a specific example. The formula description is as follows:

$$D_i(h_1, h_2) = \max_{s_1 \in h_1(a_i), s_2 \in h_2(a_i)} L(s_1, s_2), \tag{2}$$

where s_1 and s_2 are the corresponding position in h_1's and h_2's sense squares of a_i. $L(s_1, s_2)$ is defined as

$$L(s_1, s_2) = \begin{cases} 1, & P_{s_1} \neq P_{s_2}, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

where P_{s_1} and P_{s_2} mean the pieces in positions s_1 and s_2.

In this sense, $D_i(h_1, h_2) = 0$ means states h_1 and h_2 cannot be distinguished by sense action a_i. Let H_{D_i} be the maximum subset of the current information set which contains the largest number of indistinguishable states given the sense action a_i. H_{D_i} satisfies the following three constraints:

(1) $H_{D_i} \subseteq I$.
(2) $\forall h_1, h_2 \in H_{D_i}, D_i(h_1, h_2) = 0$.
(3) $\forall h_i \in H_{D_i}, \sum_{h_j \in H_{D_i}} D_i(h_i, h_j) > 0$.

We define that

$$C(a_i, I) = |H_{D_i}|. \tag{4}$$

Note that, in some cases, there may exist more than one maximum subset and the indistinguishable states they contain are different, but values of $C(a_i, I)$ are the same. In this way, given sense action a_i and current real-time information set I_k, the sense efficiency can be described by the following heuristic function:

$$H(a_i, I_k) = n - C(a_i, I_k) = n - C(a_i, I). \tag{5}$$

```
Input: Information set of agent in turn $k$: $I_k$, The set of legal sense actions: $A_s$
Output: Sense action: $a_s$
(1)  for each $h_j$ in $I_k$ do
(2)      Perform every legal move action for $h_j$ to get successors
(3)  end for
(4)  Combine all successors to update $I_k$
(5)  Let $T = 0, a_s =$ None
(6)  for each $a_i$ in $A_s$ do
(7)      for each two states $(h_1, h_2)$ in $I_k$ do
(8)          Calculate $D_i(h_1, h_2)$ using formula (2)
(9)  end for
(10) Find $H_{D_i}$ and calculate $C(a_i, I_k)$ using formula (4)
(11) Calculate $H(a_i, I_k)$ using formula (5)
(12) if $T < H(a_i, I_k)$ then
(13)         $a_s = a_i$
(14)         $T = H(a_i, I_k)$
(15)     end if
(16) end for
(17) return $a_s$.
```

Algorithm 1: HSUC algorithm.

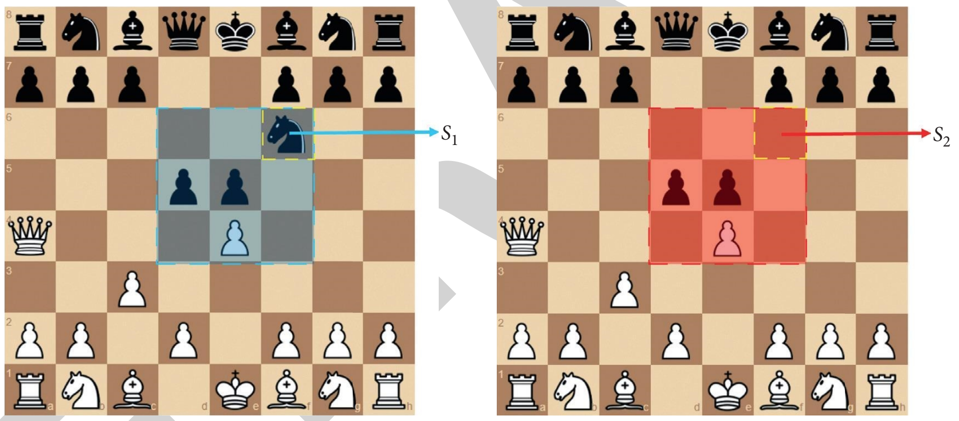

Figure 3: An example of evaluating sense efficiency. The two chess boards above denote two different states h_1 and $h_2$in the current information set. As shown in the shaded part of the figure, when sense action a_i is adopted, a difference between the upper right corners s_1 and s_2 appear. In this case, $L(s_1, s_2) = 1$ and $D_i(h_1, h_2)$ is set to value 1.

By using the heuristic function $H(a_i, I_k)$ to evaluate g_k, sense action can be searched in $A_s(I_k)$ as follows:

$$a = \arg \max_{a_i \in A_s(I_k)} H(a_i, I_k). \tag{6}$$

By now, we have presented the details of HSUC method which adopts a heuristic searching approach to minimize the information sets' scale. The whole algorithm is shown in Algorithm 1.

3.2. Foundation of the RBC Game System. In this section, we will introduce the framework of our RBC game system incorporating the HSUC method. As shown in Figure 4, our architecture contains two main parts, HSUC for sense strategy and MCTS for move strategy.

When it is our turn in the RBC game, for example, at step t, firstly, we keep an information set of step $(t-1)$ which contains all possible board states formed by our last move action. Then, we simulate all legal opponent's move actions on these board states to form the initial information set of step t. Generally, the scale of the information set will increase rapidly at the rate of dozens of times in this stage. And then, we apply our sense action provided by the HSUC algorithm. Powerful sense strategies will effectively eliminate impossible states as many as possible to get a reduced information set. At last, each remaining state will be solved as a root node by MCTS method and all of the returned solutions will be counted and the move strategy is determined by the statistics result. MCTS consists of four steps per iteration generally: selection, expansion, simulation, and backpropagation [13]. To control the iteration time, we use Stockfish to speed up

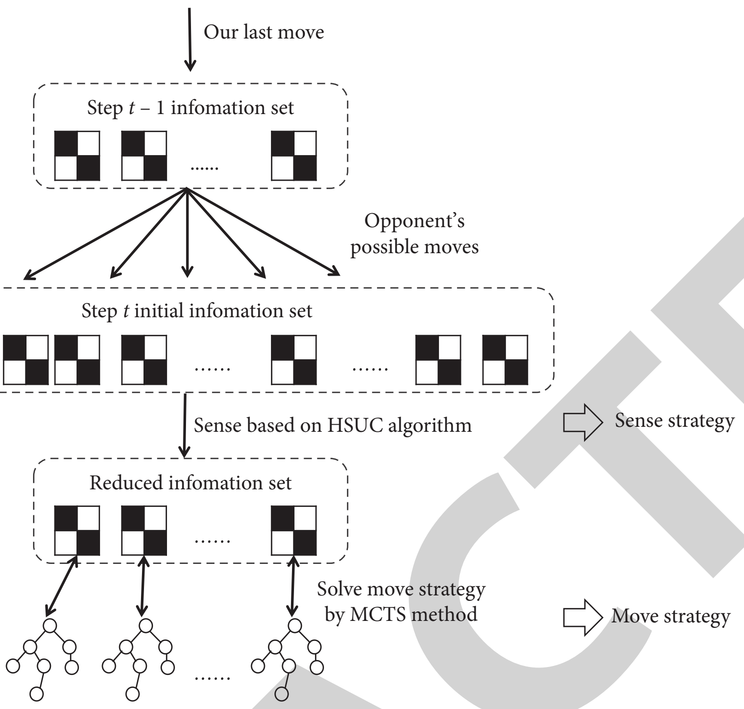
Our last move
Step $t-1$ infomation set
Opponent's possible moves
Step t initial infomation set
Sense based on HSUC algorithm
Sense strategy
Reduced infomation set
Solve move strategy by MCTS method
Move strategy

FIGURE 4: Architecture of our RBC game.

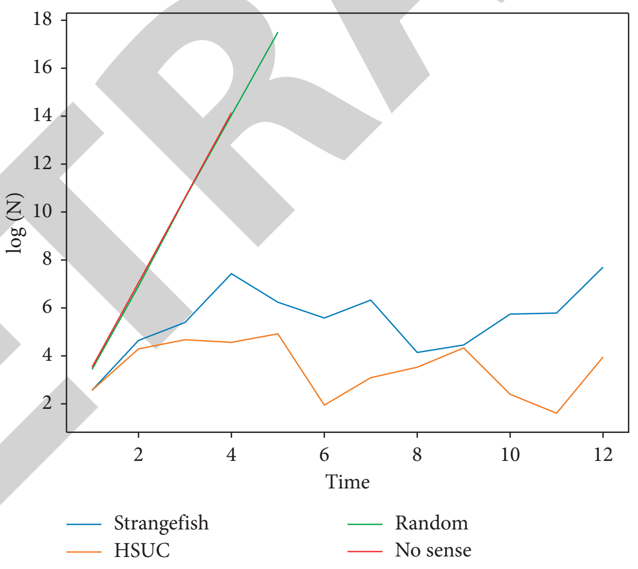

FIGURE 5: Result of different sense strategies. N is the number of states in the information set of RBC.

the termination of iterations and we constrain the depth of iterations based on the remaining time.

4. Experiments

In this section, we evaluate the effectiveness of HSUC and the RBC game-solving framework mentioned in the previous section. During the experiment setting, we choose one agent from the NeurIPS 2019 tournament, Strangefish (https://github.com/ginop/reconchess-strangefish), as a comparison baseline. Strangefish ranks the first place in the tournament, which makes the comparison in the experiments much more convincing. We conduct two experiments to verify the effectiveness of our proposed sense method HSUC (Section 4.1) and the performance of the overall RBC Game System (Section 4.2). The experiments are based on the package provided by the tournament's organizer, and the agents we implement all comply with the competition rules and restrictions.

4.1. Performance of HSUC in RBC Sense Phase. To illustrate the growth rate of the number of states in RBC's information set and the importance of a good sense method in solving

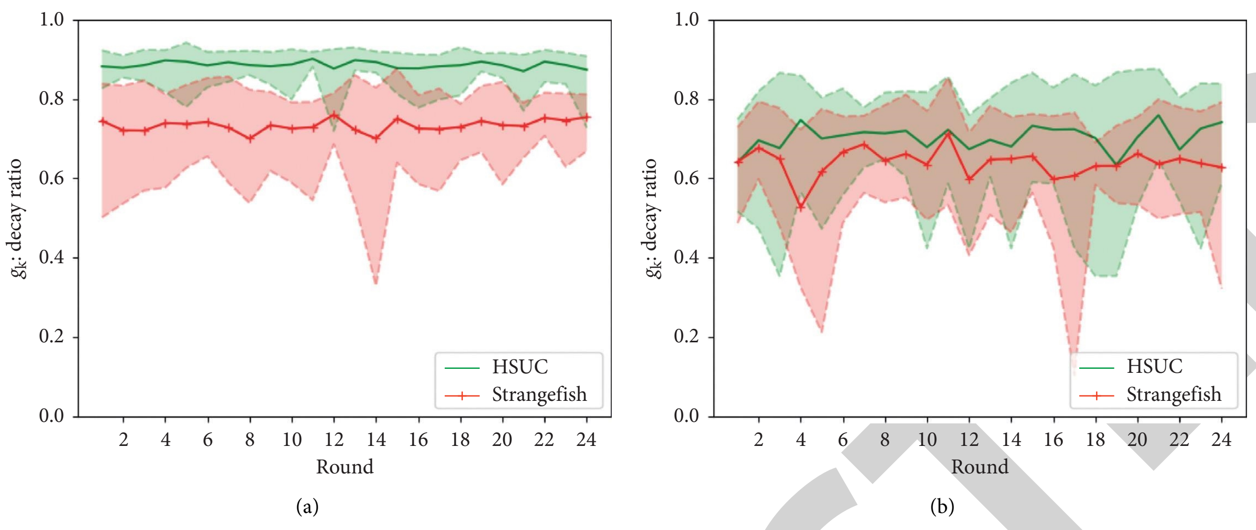

Figure 6: Efficiency of HSUC compared with sense method of Strangefish. Curves show the decay ratio g_k of the game during our agent with HSUC and Strangefish playing against different bots, respectively. The curve of our agent is higher than Strangefish in a large margin, which means that HSUC can maintain an obviously higher decay ratio of game uncertainty than Strangefish. (a) Against Random bot. (b) Against Trout bot.

IIGs like RBC, we conducted a comparative experiment of different sense methods firstly. In the experiment, agents with different sense methods play against the same opponent agent. During the game, the number of states in the real-time information sets after each sense action of each agent is tracked to obtain the experiment result.

As shown in Figure 5, the number of states in the information set of RBC increases exponentially without sense, and the problem of the exponential explosion of information sets can only be slightly alleviated with random sense actions. HSUC from our system and the sense method of Strangefish both perform better than the other two sense methods. We can conclude that the use of good sense methods can greatly reduce the scale of information set in RBC game.

The second experiment aims to verify the empirical advantages of HSUC. We let our RBC game system with HSUC compete with the baseline system at the platform of the NeurIPS 2019 tournament for 10 batches of games to obtain a statistical result. Each batch contains 24 rounds of games and each agent plays 12 rounds as black and 12 as white.

A robust sense method should satisfy the requests of efficiency and stability at the same time. First, decay ratio g_k mentioned in Section 3.1 is used to describe efficiency. The higher decay ratio denotes the method can reduce more impossible states by sensing. Second, the performance of the sense method should not fluctuate too much when facing opponents from different levels, which can be evaluated by the average scale of real-time information sets. The average scale of real-time information sets of turn j is N^j, which is calculated as follows:

$$\operatorname{avg} N^j = \sum_{i=1}^{T} \frac{N_i^j}{T}, \tag{7}$$

where N_i^j denotes the average number of states of turn j in batch i and T is the total number of batches. We use the same experimental settings as above for the experiments.

The sense strategy of the Strangefish system is employed by scoring each move action to predict for sense area. The Strangefish method picks up the move with the highest score as the most likely action of the opponent and selects sense area based on the move action. It is similar to the method we introduce in Section 3.1.

The specific implementation of the experiment is to employ another agent as the opponent of our agent with HSUC and Strangefish, respectively, to collect data for calculating the indicator g_k for efficiency and $\operatorname{avg} N^j$ for stability during the game.

In Figure 6, we compare HSUC and the sense method of Strangefish by the decay ratio g_k. Figure 6(a) shows the result of playing against Random bot and Figure 6(b) is about the result of playing against Trout (another bot using Stockfish which performs better than random in the tournament). It can be seen that HSUC maintains a better performance against different bots than sense method of Strangefish. Moreover, HSUC shows more obvious advantages when playing against agents with a higher degree of randomness by reducing no less than 90% of uncertainty. The effective decay of uncertainty will bring great advantage to follow-up and can avoid the risk of failure due to timeout in game which is suffered by the agents maintaining a large number of states in the real-time information sets.

Figure 7 shows the average scale $\operatorname{avg} N^j$ of the real-time information sets of HSUC and Strangefish. The curves in the figure obviously indicate that our method performs better on managing scale of information sets than the method of the Strangefish against both of the rivals. In Figure 7(b), the maximal average scale of information sets of Strangefish even

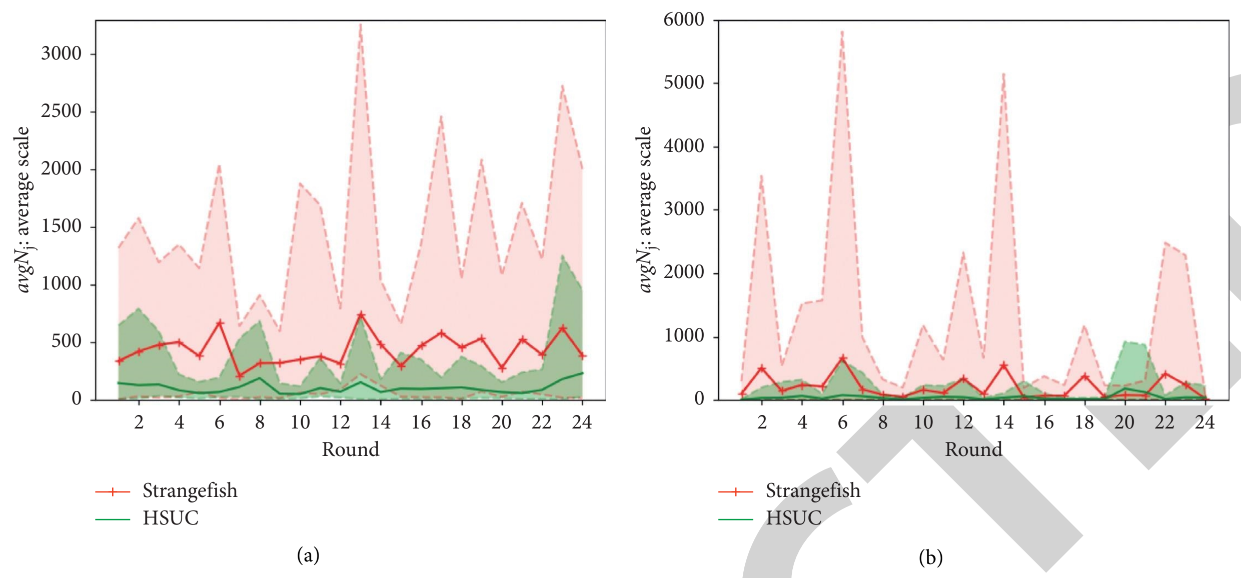

Figure 7: Stability of HSUC compared with the sense method of Strangefish. Curves show the average scale avg N^j of game during our agent and Strangefish playing against different bots, respectively. The curve of HSUC is lower and smoother than Strangefish plainly, which means that HSUC can control the scale of information sets more stably than Strangefish. (a) Against Random bot. (b) Against Trout bot.

Table 3: Uncertainty management rank.

Bot name	Institution	Median state space size
HSUC	HIT (OUR)	12
Oracle	JHU/APL	13
Wbernar5	JHU	18
La Salle Bot	La Salle University	19
MBot	SRC/Leela Chess Dev	574
Random	JHU/APL	50k+

reached 6000 while that of HSUC is about 1000. The number of states of HSUC fluctuates smoothly while that of Strangefish fluctuates violently. Besides, considering Figure 6, we can conclude that the performance of HSUC is pretty stable for each turn and against different opponents from different levels.

By the way, the result of the NeurIPS 2019 tournament can also be a reference for the effectiveness of the sense method. As shown in Table 3, our sense method performs best on uncertainty management rank (https://slideslive.com/38923177/reconnaissance-blind-chess-competition).

4.2. Performance of Our RBC Game System in NeurIPS 2019 Tournament. In the NeurIPS 2019 tournament, each agent will fight against all the other opponents in turn by 24 rounds, and each agent begins with a cumulative 15-minute clock to make all their actions including sense and move. Our agent A_bot, constructed with HSUC for sensing information and MCTS + UCT for move selection which incorporates new evaluation function Stockfish, achieves good result against many competitive opponents (such as agents from Microsoft and Google). For more details, please check here (https://rbc.jhuapl.edu/tournaments/26).

5. Conclusion

This paper introduces a novel method of uncertainty management in IIGs called HSUC. HSUC adopts a heuristic search process to guide sense actions to reduce the environment uncertainty of IIGs like RBC by minimizing the number of possible states in the real-time information sets. That is, HSUC can help agents to well understand the environment under imperfect information which enhances the effectiveness of game strategies. Furthermore, a viable RBC game system is realized by combining HSUC for sensing information and MCTS + UCT for selecting move actions. The experiments about HSUC and the RBC game system show that the scale of information sets is reduced effectively and efficiently through our method, providing convincing verification for the superiority of our method in terms of the uncertainty management in IIGs. In the future, we will conduct further research on factors affecting the uncertainty of the game environment and enrich the methods family of uncertainty management in IIGs.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding this paper.

Acknowledgments

The authors thank all the researchers in this field. This research was supported by Pingan-Hitsz Intelligence Finance Research Center, Key Technology Program of Shenzhen, China (no. JSGG20170823152809704), Key Technology

Program of Shenzhen, China (no. JSGG20170824163239586), and Basic Research Project of Shenzhen, China (no. JCYJ20180507183624136).

References

[1] J. Newman, C. L. Richardson, and S. M. Kain, “Reconnaissance blind multi-chess: an experimentation platform for ISR sensor fusion and resource management,”in Signal Processing, Sensor/Information Fusion, and Target Recognition XXV, International Society for Optics and Photonics. SPIE, pp. 62–81, Bellingham, WA, USA, 2016.

[2] R. Coulom, “Efficient selectivity and backup operators in monte-carlo tree search,” in *Computers and Games*, pp. 72–83, Springer Berlin Heidelberg, Berlin, Germany, 2007.

[3] M. Sustr, V. Kovarik, and V. Lisy, “Monte Carlo continual resolving for online strategy computation in imperfect information games,” 2018, https://arxiv.org/pdf/1812.07351.pdf.

[4] L. Kocsis and C. Szepesvári, “Bandit based monte-carlo planning,” in *Machine Learning: ECML 2006*, pp. 282–293, Springer Berlin Heidelberg, Berlin, Germany, 2006.

[5] D. E. Knuth and R. W. Moore, “An analysis of alpha-beta pruning,” *Artificial Intelligence*, vol. 6, no. 4, pp. 293–326, 1975.

[6] D. Silver, A. Huang, C. J. Maddison et al., “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[7] N. Brown and T. Sandholm, “Superhuman ai for heads-up no-limit poker: libratus beats top professionals,” *Science*, vol. 359, no. 6374, pp. 418–424, 2018.

[8] N. Brown and T. Sandholm, “Superhuman AI for multiplayer poker,” *Science*, vol. 365, no. 6456, pp. 885–890, 2019.

[9] C. E. Shannon, “XXII. programming a computer for playing chess,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 41, no. 314, pp. 256–275, 1950.

[10] J. Markowitz, R. W. Gardner, and A. J. Llorens, “On the complexity of reconnaissance blind chess,” 2018, https://arxiv.org/abs/1811.03119v2.

[11] J. J. Lu and M. Zhang, *Heuristic Search*, Springer, New York, NY, USA, 2013.

[12] M. Moravčík, M. Schmid, N Burch et al., “Deepstack: expert-level artificial intelligence in heads-up no-limit poker,” *Science (New York, N.Y.)*, vol. 356, no. 6337, pp. 508–513, 2017.

[13] C. B. Browne, E. Powley, D. Whitehouse et al., “A survey of Monte Carlo tree search methods,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.