Bridge Bidding via Deep Reinforcement Learning and Belief Monte Carlo Search

Zizhang Qiu, Shouguang Wang, *Senior Member, IEEE*, Dan You, *Member, IEEE*, and MengChu Zhou, *Fellow, IEEE*

***Abstract*—Contract Bridge, a four-player imperfect information game, comprises two phases: bidding and playing. While computer programs excel at playing, bidding presents a challenging aspect due to the need for information exchange with partners and interference with communication of opponents. In this work, we introduce a Bridge bidding agent that combines supervised learning, deep reinforcement learning via self-play, and a test-time search approach. Our experiments demonstrate that our agent outperforms WBridge5, a highly regarded computer Bridge software that has won multiple world championships, by a performance of 0.98 IMPs (international match points) per deal over 10 000 deals, with a much cost-effective approach. The performance significantly surpasses previous state-of-the-art (0.85 IMPs per deal). Note 0.1 IMPs per deal is a significant improvement in Bridge bidding.**

***Index Terms*—Contract Bridge, reinforcement learning, search.**

I. Introduction

GAMES have consistently served as fertile ground for the development and evaluation of artificial intelligence (AI) systems. Particularly, in the realm of perfect information games, recent research endeavors have yielded notable breakthroughs. Noteworthy instances include AlphaGo [1] and AlphaGo Zero [2], both of which have demonstrated superior performance over human experts in the game of Go. Expanding their prowess, AlphaZero [3] has emerged triumphant against world champions not only in chess but also in shogi.

Recent years have witnessed substantial research on imperfect information games. Impressive progress has been achieved in various complex imperfect information games. Li *et al.* [4] leverage techniques encompassing global reward prediction, oracle guiding and parametric Monte-Carlo policy adaption to achieve superhuman level in four-player game Mahjong. Libratus [5] and Pluribus [6] outperform human players in no-limit Texas Hold'em with counterfactual regret minimization. Zha *et al.* [7] employs deep Monte-Carlo methods to achieve remarkable performance in three-player game DouDizhu. Yang *et al.* [8] achieves impressive results in DouDizhu with a perfect-training-imperfect-execution framework and proximal policy optimization algorithm.

Search holds significant importance in the context of imperfect information games. Notably, SPARTA [9] and LBS (learned belief search) [10] excel in the cooperative game Hanabi by effectively employing search techniques alongside a blueprint policy. ReBeL [11] integrates reinforcement learning with search methodologies to achieve a remarkable superhuman performance in heads-up no-limit Texas Hold'em poker.

In the past few decades, multi-agent frameworks have captured considerable attention [12]–[14], propelled by the evolution of automated control, computing technology, and the burgeoning field of artificial intelligence.

This work is dedicated to creating an AI agent specifically tailored for an imperfect information multi-agent game named Contract Bridge, or simply Bridge. Bridge poses a formidable challenge to AI researchers due to its intricate nature. It is a trick-taking card game that involves four players forming two partnerships. The game unfolds in two distinct phases: bidding and playing. During the bidding phase of Bridge, players sequentially make calls, which include Pass, Double, Redouble, and a bid. A bid includes a target level of tricks and a trump suit (or a declaration of no trump). These bids are required to be in ascending order, primarily determined by the number of tricks and then by the trump suit. The final bid becomes the contract and specifies the declarer, trump suit, and target number of tricks for the ensuing playing phase. In the subsequent playing phase of Bridge, players take turns strategically playing cards from their hands in an effort to maximize their accumulation of tricks. The objective is to win as many tricks as possible. The final score is calculated based on the number of tricks made and the target level of the contract. A contract with a higher level yields a significant bonus score if it is successfully fulfilled. A detailed introduction to the rules of Bridge is given in Appendix A.

Playing is relatively less complex than bidding for Bridge AI programs to handle. In 1998, GIB [15] won the 12th in a par contest, a competition without a bidding phase. Recently, Jack [16] and WBridge5 [17] have clinched numerous titles in the World Computer Bridge Championship (WCBC) [18], signifying their remarkable prowess. In 2022, Nook [19] defeated 8 world Bridge champion players in a competition focused

Manuscript received November 24, 2023; revised March 6, 2024; accepted April 16, 2024. Recommended by Associate Editor Jiacun Wang. *(Corresponding authors: Shouguang Wang and MengChu Zhou.)*

Citation: Z. Qiu, S. Wang, D. You, and M. C. Zhou, "Bridge bidding via deep reinforcement learning and belief Monte Carlo search," *IEEE/CAA J. Autom. Sinica*, vol. 11, no. 10, pp. 2111–2122, Oct. 2024.

The authors are with the School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China (e-mail: 21020090023@pop.zjgsu.edu.cn; wangshouguang@zjgsu.edu.cn; youdan@zjgsu.edu.cn; zmc@zjgsu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2024.124488

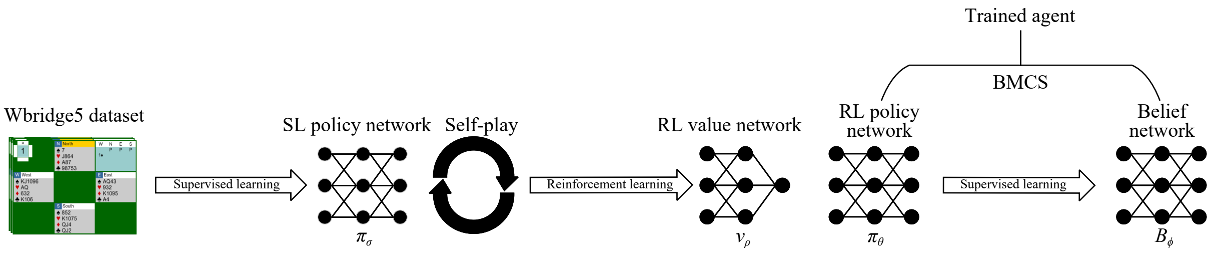


Fig. 1. Training pipeline. A supervised learning (SL) policy network π_σ is trained to predict expert calls in WBridge5 dataset. A reinforcement learning (RL) policy network π_θ is initialized to the SL policy network π_σ and then is enhanced by policy gradient through self-play. In the meantime, an RL value network v_ρ is trained to predict the expected game outcome at a specific game node. A belief network B_ϕ is trained to predict cards of other players through supervised learning on deals played by π_θ. Finally, a belief Monte Carlo search (BMCS) is proposed by combining the belief network and the RL policy network for deciding a call to be made.

solely on the playing phase of Bridge, showing superhuman performance in Bridge playing.

The bidding phase presents a higher level of challenge and complexity. Within this phase, players are privy only to their own cards (private information) and the bidding history (public information). Successful navigation of a bidding phase necessitates players to engage in information exchange and search for the most suitable contract. At times, players must also deduce and interfere with the communication between their opponents in order to make informed decisions.

In a recent study, Lockhart *et al.* from DeepMind used a combination of imitation learning, search methods, and policy iteration to train Bridge bidding agents and achieved state-of-the-art (SOTA) performance [20]. They started by imitating deals played by WBridge5 to build a strong initial policy network. Then, they improved this network through a combination of policy iteration and search methods. Optionally, they applied a test-time search to further boost the performance of the agent. We will refer to the agent using a test-time search as PI-Search and the one relying only on a single policy network trained by policy iteration as PI.

Due to a large number of possible hidden card combinations, approximately 8×10^{16} from a single player's perspective, employing uniform sampling in PI-Search makes it challenging to generate the deals that closely resemble the current one, especially when the bidding history is long. Therefore, we propose to integrate the policy network with a belief network, which is trained to predict cards of other players. This integration aims to sample deals during the search process, which closely resemble the actual deal, thereby enhancing the overall performance of the search.

The main contributions of this article are:

1) A cost-effective deep reinforcement learning approach for training a Bridge bidding agent. Unlike the training procedures for PI-Search and PI, which utilized thousands of virtual machines for acting, our method achieves better results using a single machine with multiple acting threads.

2) A novel search-based method which integrates a belief network to predict cards of other players and a policy network to evaluate candidate actions. This method improves the performance of agent at test-time further than the search method used by PI-Search.

In this work, we employ a multi-stage machine learning pipeline, as shown in Fig. 1, to train an agent for Bridge bidding. Our approach begins by initializing a policy network π_σ, which is used to make a call, through supervised learning (SL) from the deals played by WBridge5, thereby establishing a strong initial policy. Subsequently, we use a distributed deep reinforcement learning (RL) algorithm through self-play to train an RL policy network π_θ and a value network v_ρ. Moreover, we train a belief network B_ϕ to predict cards held by other players and introduce a search algorithm that combines B_ϕ with π_θ during testing for deciding a call to be made. The trained agent outperforms WBridge5, an award-winning Bridge software, by a large margin of 0.98 IMPs (international match points) per deal, with a 0.05 standard error of the mean, which is significantly better than the previous SOTA (0.85 IMPs per deal). It is important to point that 0.1 IMPs per deal is a huge improvement when comparing with player with similar game level in Bridge bidding [21].

II. Background

A. Imperfect Information Extensive Form Games

Many card games, such as Bridge and DouDizhu, can be defined as imperfect information extensive form games. An imperfect information extensive form game consists of the following components:

1) C: The set of players involved in the game, denoted as $C=\{1,2,\dots,c\}$.

2) $\mathcal{A}$: The set of all actions in action space.

3) $\mathcal{H}$: The set of all nodes in the game tree.

4) $\mathcal{Z}$: The set of all terminal nodes in the game tree.

5) $\mathcal{T}(h,a)$: The transition function leading node $h\in\mathcal{H}$ to a successor node after selecting action $a\in\mathcal{A}$.

6) $r^c(h)$: The reward function assigning a reward to each player c at a game node h.

Players act according to a policy $\pi(a|h)$ that outputs the probability distribution over all legal actions $a\in\mathcal{A}$. For each player $c\in C$, the objective is to maximize the expected reward at a certain game node h, i.e.,

$$R_c=\mathbb{E}_{h\sim\pi}[r^c(h)],\ \ h\in\mathcal{H}.$$

B. WBridge5

WBridge5, a Contract Bridge software developed by Yves Costel, has emerged as a prominent contender in the domain.

Its accomplishments include clinching victory in the World Computer Bridge Championship (WWBC) on six separate occasions. Moreover, WBridge5 serves as a consistent benchmark, having been employed as a reference point for assessment in earlier researches [20], [22], [23].

Although WBridge5 is a close source software, accessibility is facilitated through the Blue Chip Protocol—a communication protocol employed within WWBC. This protocol enables interaction with WBridge5 via TCP connection.

In this work, our evaluation methodology involves pitting our agent against WBridge5, a strategic approach aimed at gauging the effectiveness and strength of our agent.

III. Supervised Learning of a Policy Network

In the initial stage of training, we employ a supervised learning approach to obtain a policy network $\pi_\sigma(a|h)$, which takes a representation of game node h as its input (see Appendix B), outputting a probability distribution over all possible calls a. The training adopts a dataset of approximately 1 million deals played by WBridge5 [20] using the Standard American Yellow Card (SAYC) bidding system, a widely adopted system in Bridge tournaments and online gameplay. The training objective is to determine weights σ such that policy network π_σ can imitate the decision-making patterns demonstrated by experts.

Our policy network π_σ is a 4-layer multilayer perceptron (MLP) with 2048 neurons in each layer. It is trained on node-action pairs (h, a), by using gradient descent to maximize the log likelihood of action a

$$\Delta\sigma \propto \frac{\partial \log(\pi_\sigma(a|h))}{\partial\sigma}.$$

Also, the GELU activation [24] and a softmax output are used to get probability distribution over all calls a.

To be clear, we call the policy network obtained in this stage an SL policy network, denoted as π_σ.

IV. Deep Reinforcement Learning of a Policy Network

The second stage of the training pipeline focuses on enhancing the policy network through deep RL. We call the policy network obtained in this stage an RL policy network, denoted as π_θ.

In RL, the state space encompasses the entirety of potential bidding scenarios, which fluctuates based on factors such as vulnerability, bidding history, and the cards held. The action space comprises all calls during the bidding phase.

At the beginning, π_θ is initialized to π_σ. Then we use a self-play method with policy gradient [25] to train π_θ. Following previous work [23], [26], we use IMP as the reward in the RL stage. In this scheme, each deal is played twice. Notably, the agents occupy the North-South positions in the first game and subsequently assume the East-West roles in the second game. Upon the conclusion of both games, duplicate scores are calculated through the utilization of pre-computed double dummy results using double dummy solver (DDS) [27]. These scores are then converted into IMP and subsequently normalized to fall within the range of [−1, 1]. This normalized value of IMP represents the game outcome and functions as the only sparse reward r_t in the RL stage. This reward setting may mitigate the influence of random card distribution variations.

In addition, we train a value network $v_\rho(h)$ with weights ρ to estimate the value function

$$v^\pi(h) = \mathbb{E}[r_t|h_t = h, a_{t\ldots T} \sim \pi].$$

It predicts the expected reward, i.e., game outcome, at node h in the case that all the players use policy π at time step t. The value network shares a similar architecture with the policy network, with the distinction that it outputs a scalar value representing the predicted reward, instead of action probabilities. To overcome the uncertainty of imperfect information, the input features to the value network during training include not only those used by the policy network, but also the hidden information, i.e., the cards held by the other three players, which is an employed approach in StarCraft [28] and DouDizhu [8].

We utilize a distributed setting of deep RL. Multiple actor threads run in parallel to generate trajectories and send transitions into a prioritized replay buffer [29]. The transitions are prioritized by using the advantage value

$$A(h_t, a_t) = r_t - v_\rho(h_t)$$

which is the difference between the ground truth reward and the predicted one. In actor threads, at each non-terminal node h, we use policy network π_θ to obtain the probability distribution of all actions $\pi_\theta(\cdot|h)$ and use value network v_ρ to obtain the reward estimation $v_\rho(h)$. Then we sample an action a from $\pi_\theta(\cdot|h)$ and store the transition $(h, a, v_\rho(h), \pi_\theta(\cdot|h))$ in a local buffer $\mathcal{B}_L$. When a terminal node is reached, we compute duplicate scores of 2 tables using pre-computed double dummy results and convert them into IMP as reward r. Then we update the transitions in local buffer $\mathcal{B}_L$ and store them in a shared replay buffer $\mathcal{B}$.

Asynchronously, a learner thread samples mini-batches from the replay buffer and updates the policy network and value network based on the A2C algorithm [30]. At each time step t, the policy network is updated to maximize the expected reward with the advantage, entropy regularization, and importance sampling

$$\Delta\theta \propto \frac{\pi_\theta(a_t|h_t)}{\pi_{\theta'}(a_t|h_t)} \nabla_\theta \log \pi_\theta(a_t|h_t) A(h_t, a_t) + \beta \nabla_\theta H(\pi_\theta(\cdot|h_t))$$

where $\pi_{\theta'}$ is the former policy distribution saved in the replay buffer, H represents the entropy of the policy pertaining to node h_t, and hyperparameter β is the weight of entropy, which controls the extent of exploration. The value network is updated to minimize the mean squared error (MSE) between the predicted reward $v_\rho(h_t)$ and the ground truth one r_t

$$\Delta\rho \propto \frac{\partial (r_t - v_\rho(h_t))^2}{\partial\rho}.$$

Pseudocodes of actor and leaner threads are provided in Appendix C.

V. Searching With a Belief Network

Under some specific conditions, an agent could find itself confronted with multiple potential actions strongly advocated by policy network π_θ. However, it is essential to recognize that the action possessing the highest probability might not necessarily yield the most substantial reward. To address this issue, previous work has applied search methods to find the optimal action in such a case.

GIB [15] utilizes a technique known as Borel search. It begins by forming a set of deals that match the ongoing bidding sequence. Then, it calculates the score for each potential action in each deal using double dummy results. The action with the highest cumulative score is chosen as the final decision.

The extended Borel search [20] addresses the challenge of low similarity between a sampled deal and actual one by filtering the deals based on the probability of actions taken in the bidding history, leveraging a policy network. However, due to the vast number of potential hidden card combinations, i.e., around 8×10^{16} from the perspective of a single player, uniform sampling makes it difficult to generate deals closely resembling the current deal, particularly when the bidding history is long.

To tackle this challenge, we introduce a novel approach called belief Monte Carlo search (BMCS). It begins by training a belief network $B_\phi(b|h)$, with weights ϕ, to predict the belief b, i.e., the cards held by the remaining three players from the perspective of the searching player at game node h. The belief network shares an architecture similar to the RL policy network, taking the same input features as the latter, but differs in its last layer, where a sigmoid function is used to output a belief distribution, i.e., a probability distribution of the cards held by other three players. In more detail, the output of our belief network is a 156-element vector, where each 52 elements correspond to the cards of a player and every element indicates the probability of whether the card is held by the player. We train the belief network by using SL on node-belief pairs (h, b) generated through self-play on random deals, utilizing the RL policy network π_θ. The belief network is optimized by minimizing a cross-entropy loss.

The search algorithm for node h starts by selecting candidate actions using π_θ. We only consider actions with a probability higher than a pre-set probability threshold $p_{\min}$. Subsequently, we sample possible deals consistent with the cards held by the searching player from the belief distribution $B_\phi(b|h)$. Then a filter procedure similar to the extended Borel search is employed. We filter the deals not closely similar to the actual one by the probability of actions in bidding history using the RL policy network π_θ. For each action a' in node h' of bidding history, we eliminate the sampled deal by probability

$$1-\pi_\theta(a'|h').$$

For the remaining deals, we use π_θ to perform rollouts for each action. The scores for actions are computed based on the results of DDS. The search ends when the number of rollouts performed reaches a pre-set threshold $R_{\max}$ or the number of sampled deals reaches a pre-set threshold $P_{\max}$. Once the search is complete, the algorithm selects the action with the maximum cumulative score if the number of rollouts performed is larger than a pre-set threshold $R_{\min}$. Otherwise, the algorithm falls back to the greedy action computed according to the policy distribution $\pi_\theta(\cdot|h)$. This choice is made to mitigate the impact of randomness when the number of deals used for rollouts is relatively small. Details and pesudocode of BMCS is provided in Appendix D.

VI. Experiments

A. Supervised Learning

We pre-process the WBridge5 self-play dataset to form node-action pairs (h, a), and subsequently partition it into training and test sets. For each training step, we extract a mini-batch $\{h^i,a^i\}_{i=1}^m$ of m samples from the dataset and update the network weights σ by minimizing a cross-entropy loss

$$\mathcal{L}_{\pi_\sigma}=-\frac{1}{m}\sum_{i=1}^{m}\log(\pi_\sigma(a^i|h^i)).$$

We employ the Adam optimizer [31] with a learning rate of 1×10^{-4} and mini-batch size $m=512$. Consistent with earlier studies [20], [22], we do not use auxiliary loss in our experiments.

The training takes several hours by using a single GPU. Our SL policy network achieved accuracy of 94.37% on the test set.

B. Deep Reinforcement Learning

Implementation Details: The experiments are conducted using a single Intel(R) Xeon(R) Gold 5218R CPU @ 2.10 GHz and a NVIDIA A40 GPU. Consistent with prior research [20], vulnerability settings are standardized as neither side vulnerable for all experiments. We perform a grid search [32] on hyperparameters and find the best combination. Hyperparameters used in our experiments are shown in Table I. Sync frequency in it refers to the number of updates for synchronization between the networks in actor threads and the learner.

The training process reaches convergence within a span of 3 days, completing approximately 2.5×10^6 training steps. This approach offers a cost-effective alternative compared to prior studies that utilized two GPUs for a duration of 14 days [23], or involved thousands of actor threads during the training stage [20]. The difference arises mainly because we do not perform search and rollouts in RL training but JPS [23] and PI [20] do.

To assess the progress of our training, we conducted benchmarking against π_σ at the conclusion of each training epoch. Our training curve, depicted in Fig. 2, provides insight into the evolving performance of agent. In this context, an epoch corresponds to 1000 mini-batches of training data.

Notably, our best-performing RL policy network π_θ surpassed π_σ by a significant margin of 0.87 IMPs per deal. The achieved improvement is statistically robust, indicated by a low standard error of the mean at 0.02. This outcome underscores the efficacy of our approach, showcasing the substantial advancements achieved through our reinforcement learning-based training strategy.

TABLE I
HYPERPARAMETER SETTINGS IN DEEP REINFORCEMENT LEARNING EXPERIMENT

Hyperparameter	Value
# Actor threads	12
# Environments per actor thread	150
Replay buffer capacity	80 K
Replay buffer burn-in frames	20 K
Batch size	2048
Optimizer	Adan [33]
Learning rate for π_θ	1.3×10^{-5}
Learning rate for v_ρ	1.3×10^{-4}
Priority exponent	0.8
Importance sampling exponent	0.6
Entropy weight	0.01
Discount factor γ	1
Sync frequency	10

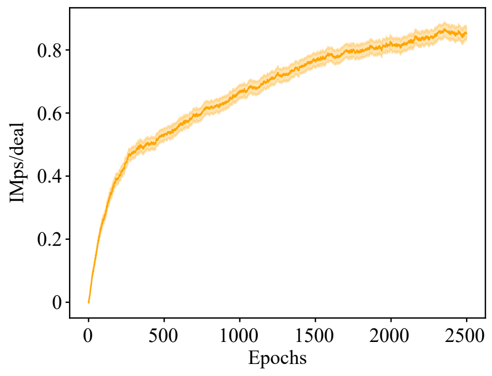


Fig. 2. Training curve of deep reinforcement learning. Each data point is evaluated on 50 k games and the shaded area is the standard error of mean.

C. Belief Network

Implementation Details: As mentioned in Section V, the training process of our belief network adopts a distributed approach. We perform a grid search on hyperparameters and find the best combination. Hyperparameters used in our experiments are shown in Table II. The belief network is trained on a GPU with approximately 3×10^5 training steps.

We assess the performance of the belief network by evaluating its quality through the computation of cross-entropy loss. This evaluation is conducted on a set of random deals played by π_θ, generating 100 000 pairs of nodes and corresponding beliefs. For each pair, we compute the cross-entropy loss by comparing the predicted belief distribution with the ground truth distribution. For comparison, we compute the cross-entropy loss for uniform sampled belief. We sample 1000 deals uniformly for each node h and compute the average cross-entropy as the result.

The results of this evaluation are illustrated in Fig. 3. The plot demonstrates the relationship between bidding length and the corresponding cross-entropy loss. Notably, we observe a consistent trend where the cross-entropy loss decreases as the bidding length becomes longer. Deals sampled uniformly consistently exhibit a high cross-entropy loss, regardless of the length of the bidding history. This observation suggests that the belief network tends to provide more accurate predictions as the bidding history becomes more extensive.

TABLE II
HYPERPARAMETER SETTINGS IN SL FOR BELIEF NETWORK

Hyperparameter	Value
# Actor threads	6
# Environments per actor thread	100
Replay buffer capacity	80 K
Replay buffer burn-in frames	20 K
Batch size	256
Optimizer	Adam [31]
Learning rate for B_ϕ	3×10^{-4}
Priority exponent	0

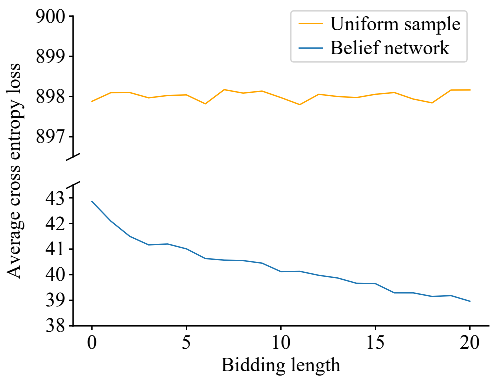


Fig. 3. Cross entropy loss of belief network with different bidding length.

D. Final Performance

To evaluate the strength of our trained agent, we run a tournament against WBridge5, which is a software that has won the World Computer Bridge Championship six times and has served as a comparator in previous work [20], [22], [23], [26].

In the tournament, we consider not only the complete version of our trained agent that we name RL-BMCS agent, but also an intermediate version that we name RL agent. The RL-BMCS agent decides an action by performing BMCS on the basis of the RL policy network, while the RL agent simply chooses an action with the highest probability output by the RL policy network. We conduct a series of 10 000 deals pitting each of the two agents against WBridge5, utilizing the IMP metric for evaluation. Their performance is shown in Table III, where the results of other trained agents, including PI, PI-Search, JPS [23], Simple [26], and DNNs (deep neural network) [22], are presented as well. We can see that our RL-BMCS agent beats WBridge5 by a large margin of 0.98 IMPs per deal, surpassing all the other agents, particularly including PI-Search agent which also utilizes a test-time search method. It is pointed out that a gain of 0.1 IMPs per deal is an significant improvement in Bridge [21]. Also, the improvement of performing BMCS is 0.30 IMPs per deal, surpassing the search method used by PI-Search (0.28 IMPs per deal), showing our BMCS enhances the strength of trained agent

TABLE III
IMPs Per Deal for Bridge Bidding AI Against WBridge5

Algorithm	Result
DNNs [22]	0.25
Simple [26]	0.41±0.27
JPS [23]	0.63±0.22
PI	0.57±0.05
PI-Search	0.85±0.05
RL (ours)	0.68±0.05
RL-BMCS (ours)	**0.98±0.05**

further. Even without BMCS, our RL agent outperforms WBridge5 by 0.68 IMPs per deal, superior to other agents without test-time search, i.e., PI, JPS, Simple and DNNs. It indicates that our reinforcement learning approach significantly improves the strength of the trained agent and the use of BMCS improves its strength even further.

It is noteworthy that our agents conform to the SAYC bidding system in the majority of the games, while agents JPS, Simple and DNNs do not. The conformity of our agents significantly reduces the potential for winning IMPs by confusing WBridge5 with unexpected calls, which is illegal in real-life tournaments. An in-depth analysis of our agents is provided in Section VI-F and bidding cases of our agents are provided in Appendix E.

E. Ablation Studies

We also assess variants of our agents by removing some components from the RL stage and BMCS. Specifically, we consider three variants:

1) No PR: An agent trained without a prioritized replay (PR) buffer, with all transitions sampled uniformly.

2) No HI: An agent trained without the hidden information (HI) as input to the value network v_ρ.

3) RL-MCS: An agent that performs search without utilizing the belief network B_ϕ, with deals sampled uniformly.

Table IV shows the average IMP obtained by these agents against WBridge5 over 10 000 same deals. The results demonstrate that the RL-MCS agent exhibits inferior performance in comparison to the RL-BMCS agent. This suggests that the incorporation of the belief network B_ϕ within the search algorithm enhances the strength of the agent. The no HI and no PR agents have worse performance compared to the RL agent, showing the necessity of considering additional information as input to v_ρ and the prioritized replay buffer in the RL stage.

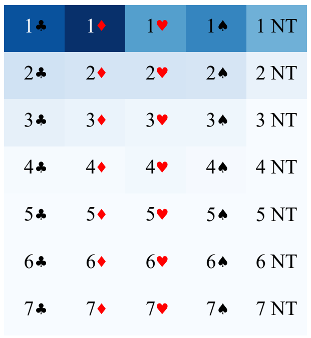
TABLE IV
IMPs Per Deal for Variants of Our Agent Against WBridge5

Algorithm	Result
No PR	0.53±0.05
No HI	0.58±0.05
RL	0.68±0.05
RL-MCS	0.80±0.05
RL-BMCS	**0.98±0.05**

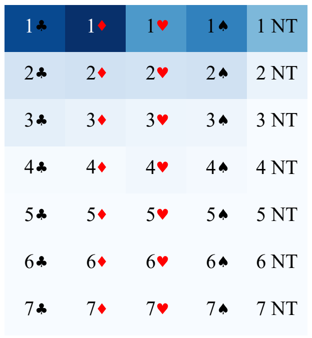
F. In-Depth Analysis

We provide further analysis of the behavior of our trained agents in the tournament, including the preference of an opening bid made, preference of the final contract, and bidding length distribution.

The opening bid, which is the first contract bid made in the bidding phase, is often regarded as a critical aspect of Bridge bidding. In the total 20 000 games played (each deal is played twice), the RL agent, makes the opening bid in 10 862 games, while the RL-BMCS agent makes the opening bid in 11 226 games. The heatmaps of different opening bids made by the two agents are shown in Fig. 4. The figure indicates that most of the opening bids are at one and two level, which is also common in real-life Bridge games.



(a)



(b)

Fig. 4. The heatmaps of opening bids made by our agents. The bid with deeper color is made more often ((a) The opening bids made by RL agent; (b) The opening bids made by RL-BMCS agent).

We then check the conformity to the SAYC convention of these opening bids. We determine conformity by conducting a straightforward comparison of high card points (HCP) and suit lengths with the SAYC guidelines. The results are shown in Table V. We have also provided the result of opening bids made by WBridge5 from all the games played. Note that this method may not provide a complete confirmation of conformity, as SAYC includes several other rules and guidelines for making opening bids that are not considered in this analysis. It shows that our agents consistently adhere to the SAYC bidding system throughout both training and search stages, reducing the chances of winning IMPs through confusion tactics against WBridge5 during evaluation.

Fig. 5 displays the results of final contracts made by our agents. The heatmap illustrates that our agents tend to favor part score contracts and game contracts, with a particular preference for 3NT, 4♡, and 4♠. Slam contracts are less frequently chosen. This behavior aligns with the typical bidding choices of human players in Bridge.

Fig. 6 shows the bidding length distribution of the games our agents played against WBridge5. Notably, most of these games require bidding length ranging from 6 to 15. Games with bidding length exceeding 20 are seldom to see. This pattern aligns closely with the characteristics of games played by human experts, as observed in a previous study [22].

VII. Discussion

In this work, we have successfully designed and imple-

TABLE V
PERCENTAGE OF FREQUENTLY MADE OPENING BID CONFORMING TO SAYC (%)

Opening bid	RL agent	RL-BMCS agent	WBridge5
1♣	74.25	61.11	89.27
1♢	76.76	73.08	88.26
1♡	73.91	63.72	80.04
1♠	75.47	65.88	79.80
1NT	76.15	69.19	85.71
2♣	25.71	26.26	32.53
2♢	92.13	92.41	94.89
2♡	94.61	90.66	93.51
2♠	94.18	86.35	95.35
2NT	53.53	48.98	82.22
3♣	33.50	28.70	41.48
3♢	49.38	36.47	74.73
3♡	50.49	41.13	49.04
3♠	65.35	50.45	55.44

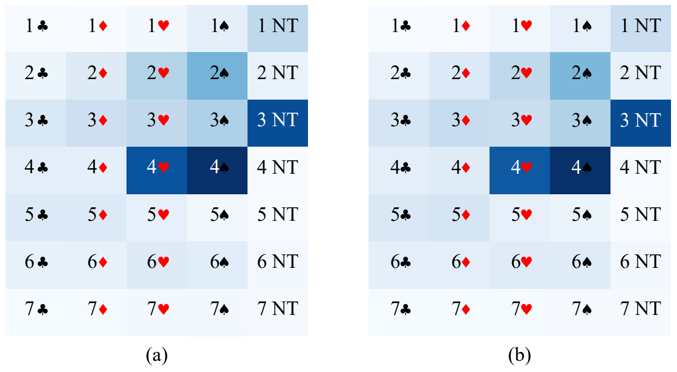


Fig. 5. Heatmaps of final contracts made by our agents. The contract with deeper color is made more often ((a) The final contracts made by RL agent; (b) The final contracts made by RL-BMCS agent).

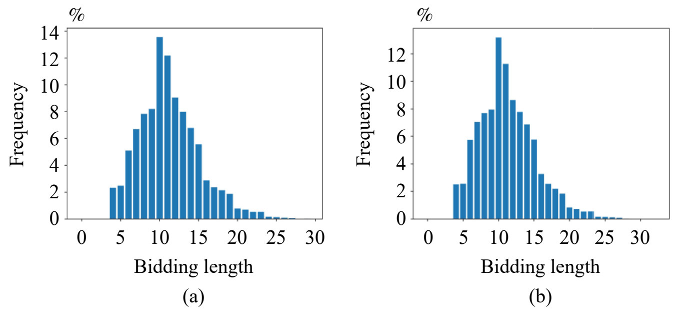


Fig. 6. Bidding length distribution ((a) The bidding length of the games played by RL agent; (b) The bidding length of the games played by RL-BMCS agent).

mented a Bridge bidding agent by leveraging the power of deep neural networks and advanced search techniques. Our approach involves a comprehensive training process including supervised learning and deep reinforcement learning. Notably, we have introduced an innovative search mechanism that combines a policy network with a belief network to effectively evaluate potential actions. As a result of these efforts, our agent has achieved SOTA performance in the realm of computer Bridge, with significantly less training budget. We are optimistic that the methodologies employed here can be adapted and applied to other imperfect information games, paving the way for future advancements in the field.

Despite the advancements achieved in our work, it is imperative to acknowledge the existing limitations. One notable limitation pertains to the assumption made during both supervised learning and deep reinforcement learning phases, where all opponents are assumed to adhere to the same bidding conventions as the agent. This might cause confusion to the agent on decision-making when confronted with opponents utilizing different bidding systems. The other limitation lies in the execution time of our BMCS approach, which takes 1 to 2 min to make a decision. It is relatively slow when compared with search methods employed in other games such as Go [1], [2] and Hanabi [9], [10], which basically take no more than 1 minute. This disparity mainly arises from the necessity of computing double dummy results. This computational overhead leads to more execution time required by our BMCS technique, posing a challenge in scenarios where real-time responsiveness is essential. Future work might solve these problems through better methods for bidding with different conventions and faster evaluation of actions.

APPENDIX A
CONTRACT BRIDGE GAME RULES

In this section, we summarize the rules of the game of Contract Bridge.

The game of Contract Bridge is played by four players seated North, East, South, and West. North and South form a partnership against West and East. Employing a standard deck comprising 52 cards, Contract Bridge adopts a classification of suits into major (♡ and ♠) and minor (♣ and ♢). Preceding the commencement of the game, each player is dealt 13 cards. The game comprises two distinct phases: bidding and playing. The final score attributed to each player emerges as a composite reflection of their performance across both phases. An example game of Bridge is shown in Fig. 7.

A. Bidding

The objective of the bidding phase is to communicate and identify the most advantageous contract. This phase follows a clockwise progression, commencing with the dealer and progressing in sequence. During their respective turns, each player is obligated to make a call. There are a total of 38 calls, and these calls can be classified into four distinct categories:

1) A bid, also referred to as a contract bid, represents a fusion of two components: the target trick level and the trump suit (or no trump). There are 35 possible bids. The set of all possible contract bids is denoted as an ordered sequence: {1♣, 1♢, 1♡, 1♠, 1NT, 2♣,..., 7NT}, with NT signifying no trump. Importantly, each subsequent contract bid must surpass its predecessor, if one exists.

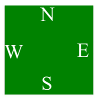
2) Double, an option exclusively accessible when the opposing team has placed the bid. Double increases the contract score and the penalty score.

3) Redouble, exclusively applicable when the contract is

North deals
None vul

♠9 6 4
♥Q
♦K J 8 7
♣K Q J 5 2

♠10 5
♥A K J 8 7
♦6 5 4
♣A 8 4



♠A K Q J 3 2
♥9 6 5 2
♦9 3
♣6

♠8 7
♥10 4 3
♦A Q 10 2
♣10 9 7 3

West	North	East	South
	1♣	3♠	Pass
Pass	Pass		

3♠ by East

Trick	Lead	2nd	3rd	4th
1. S	♣10	A	K	6
2. W	♠10	4	3	8
3. W	♠5	6	K	7
4. E	♠J	♥3	♣4	9
5. E	♥2	10	K	Q
6. W	♥7	♣5	6	4
7. W	♣8	J	♠Q	7
8. E	♥5	♦2	8	♣2
9. W	♥A	♦7	9	♦10
10. W	♥J	♦8	♦3	♣3
11. W	♦4	J	9	Q
12. S	♦A	5	K	♠A
13. E	♠2	♣9	♦6	♣Q

Made 6 — EW + 230

Fig. 7. Example of a Contract Bridge game, including both bidding and playing phase.

doubled by the opposing team. By choosing redouble, the scores are further escalated.

4) Pass, an available option in all scenarios.

The bidding phase reaches its conclusion under two distinct conditions: either when three consecutive passes occur or when all four players opt to pass at the onset of the bidding phase (with no contract bids made). In the first scenario, the ultimate contract bid made serves as the concluding contract, and the player who initially selects the trump (or NT) becomes the declarer in the playing phase. In the second scenario, all players receive zero points, and the playing phase is skipped.

B. Playing

The playing phase unfolds across 13 rounds. The partner of the declarer is the dummy, while the opposing partnership holds the designation of the defenders. Within each round, the first player sets forth a lead card, prompting the other players to follow suit if able. Alternatively, they can opt to play any card from their hand if they lack a matching suit. The initial round starts with the player positioned to the left of declarer's left—this is referred to as the opening lead. Subsequently, the cards held by dummy are revealed to all players and managed under the control of the declarer. If a trump suit card is played, the player with the highest trump wins, otherwise the player with the highest lead suit wins. The player who wins the trick leads the next round. After 13 rounds of play, each player receives a score.

For the declarer and dummy partnership, rewards are granted if their count of won tricks surpasses the contract level by an increment of 6 (for instance, a 2♣ contract necessitates the partnership to win 8 tricks). In cases where this benchmark is unmet, a negative score is incurred as a penalty.

C. Scoring

A scoring system known as vulnerability adds a layer of complexity to Bridge games. Before the game commences, vulnerability is designated for the two partnerships involved. This factor impacts the scoring dynamics by amplifying the rewards for successfully fulfilling a contract and, in a corresponding manner, intensifying the penalties incurred for falling to make the contract.

There are several rules of Bridge scoring, we only describe the duplicate scoring rules here. The score of a successfully achieved contract encompasses five distinct components:

1) Trick score, computed based on the level of the contract and trump suit. Notably, major suits and no trump contracts yield greater trick points compared to minor suits at an equivalent level. Contracts with a trick score higher than 100 are called game contracts, others are called part score contracts. Vulnerability holds no sway over the trick score, yet the double status is linked to it.

2) Overtrick score, emerging when the declarer side makes an excess of tricks beyond the stipulated count of the contract. The calculation of the overtrick score is closely linked to both the double status and vulnerability.

3) Game bonus, exhibiting a lower value for part score contracts and a significantly higher value for game contracts.

4) Slam bonus, entailing a substantial bonus computed exclusively when the level of the contract attains 6 (small slam) or 7 (grand slam). The slam bonus is solely influenced by the state of vulnerability.

5) Double bonus, which is calculated when the contract is doubled or redoubled.

The penalty for a failed contract depends on the number of undertricks, double status and vulnerability of the declarer.

In real-world tournaments, the evaluation of strength of two teams is often gauged by using international match points (IMP). The typical setup involves two identical games played at two separate tables. In each team, four players partake, with two players forming the North-South pair at one table, while the other two assume the West-East roles at the other table. Upon the completion of both game rounds, the divergence between the duplicate scores is translated into IMP, a scale that spans from 0 to 24, thus enabling a comparative assessment of the performances of teams.

Appendix B
Input of Neural Networks

We describe the input features of the neural networks. The size and simple description of features are shown in Table VI.

The input to the policy network is the imperfect feature at node h. We utilize the representation in OpenSpiel [34] as its input, which is a 480-element vector. It takes all features except the last one in Table VI as input. The first 4 bits repre-

TABLE VI
INPUT FEATURES FOR NEURAL NETWORKS

Feature	Size	Description
Vulnerability	2×2	The vulnerability of 2 partnerships
Opening pass	1×4	Whether a specific player makes an opening pass
Bid making history	35×4	Whether a specific player makes a call of a specific bid
Bid double history	35×4	Whether a specific player doubles a specific bid
Bid redouble history	35×4	Whether a specific player redoubles a specific bid
Own hand	52	cards held by current player
Hands of other players	52×3	cards held by other players

sent the vulnerability of 2 partnerships using one-hot encoding. The next 4 bits represent whether a player make a call of opening pass. The index of a specific player is relative distance clockwisely. For example, if the current player is North, and South has made a call of opening pass, the second element (start from zero) in 4 bits is set to 1 because the distance of North to East is 2. The next $4\times35=120$ bits represent the history of bids. Each bid takes 4 bits for recording which player makes a call of this bid. The bid is ordered as {1♣, 1♢, 1♡, 1♠, 1NT, 2♣,..., 7NT}. The index of a player is defined in the same way. For example, if the current player is North, West makes a call of 1NT, then the third bit in the block of 1NT is set to 1, which is the $5\times4+3=23$rd bit in the 120-bit bid history. The next two 120 bits represent whether a bid is doubled and redoubled using the same encoding method. The input to the belief network is the same as that of the policy network. The input to the value network is a 636-element vector that takes all the features in Table VI.

APPENDIX C
PSEUDOCODES OF THE ACTOR AND LEARNER THREAD

Algorithm 1 Actor Thread

```
Input: Shared prioritized replay buffer $\mathcal{B}$, local buffer $\mathcal{B}_L$, policy network $\pi_\theta$, value network $v_\rho$, network synchronization frequency $f$
for $i = 1, 2, \ldots,$ do
    if $i\% f = 0$ then
        Synchronize $\pi_\theta$, $v_\rho$ with learner thread.
    end if
    Sample a deal from training set. Form initial node $h$.
    for $t = 1, 2, \ldots, T$ do
        Sample action from policy distribution
            $a_t \sim \pi_\theta(\cdot|h_t)$
        and obtain value estimation $v_\rho(h_t)$;
        Store $(h_t, a_t, v_\rho(h_t), \pi_\theta(\cdot|h_t))$ in local buffer $\mathcal{B}_L$.
        $h_t = \mathcal{T}(h_t, a_t)$
    end for
    Compute the reward $r_t$ using pre-computed double dummy results.
    for $t = 1, 2, \ldots, T$ do
        Update transitions in $\mathcal{B}_L$ using $r_t$
        Store transition $(h_t, a_t, r_t, \pi_\theta(h_t), v_\rho(h_t))$ in $\mathcal{B}$
    end for
end for
```

Algorithm 2 Learner Thread

```
Input: Shared prioritized replay buffer $\mathcal{B}$, policy network $\pi_\theta$, value network $v_\rho$, learning rate $\eta_\theta, \eta_\rho$
for $i = 1, 2, \ldots,$ do
    Sample a mini-batch of transitions from $\mathcal{B}$
    Update $\pi_\theta$ with
        $\Delta\theta = \frac{\eta_\theta}{m}\sum_{i=1}^{m}[\frac{\pi_\theta(a_t^i|h_t^i)}{\pi_{\theta'}(a_t^i|h_t^i)}\nabla_\theta \log\pi_\theta(a_t^i|h_t^i)(r_t^i - v_\rho(h_t^i)) + \beta\nabla_\theta H\left(\pi_\theta(\cdot|h_t^i)\right)]$.
    Update $v_\rho$ with
        $\Delta\rho = \frac{\eta_\rho}{m}\sum_{i=1}^{m}\frac{\partial(r_t^i - v(h_t^i))^2}{\partial\rho}$.
    Update the priority of sampled transition using advantage
        $A(h_t, a_t) = r_t - v_\rho(h_t)$.
end for
```

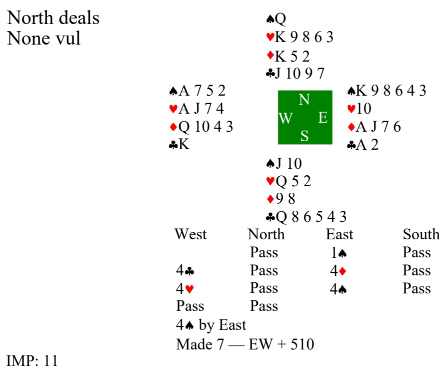
APPENDIX D
PSEUDOCODE AND DETAILS OF BELIEF MONTE CARLO SEARCH

We employ BMCS (Algorithm 3) to enhance the performance of our agent during test-time by identifying better actions. This search algorithm operates through four distinct stages.

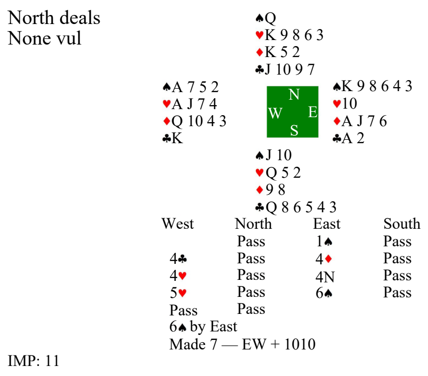
1) Candidate Action Selection: At the first stage of the search, we select candidate actions to evaluate the probability distribution over actions provided by the policy network π_θ. We select the top-k actions with a probability higher than $p_{\min}$ to ensure that these actions are likely to result in favorable outcomes.

2) Deal Sampling and Filtering: The deals are sampled by using the belief distribution output by the belief network $B_\phi(h)$. We mask out the cards held by the searching player in order to ensure that the sampled deals are consistent with the current node h. We filter the sampled deals by the probability of historical actions within these deals. Lower probabilities of actions are associated with higher chances of eliminating the corresponding deals.

3) Action Evaluation: For the remaining deals after the filtering stage, we evaluate the candidate actions by performing rollout evaluation (Algorithm 4) with the policy network π_θ. Each action first leads the game to a successor node h' by the



(a)



(b)

Fig. 8. A purely collaborative bidding case played by RL agent ((a) The table where our RL agents sit at North and South; (b) The table where our RL agents sit at East and West).

transition function $\mathcal{T}(h,a)$, actions are then selected by using the policy network until the game is terminated. The evaluation value of the candidate action based on double dummy results is accumulated.

4) Action Decision: After a specified number of deal samplings and action evaluations, the algorithm makes the final decision based on the cumulative evaluation value and the number of performed rollouts. If this number exceeds a predefined threshold $R_{\min}$, the action with the highest cumulative value is selected as the final decision. Otherwise, the algorithm chooses the greedy action from the probability distribution over actions.

Algorithm 3 Belief Monte Carlo Search

Input: Game node h, Belief model B_ϕ, policy network π_θ, minimum number of rollouts $R_{\min}$, maximum number of rollouts $R_{\max}$, maximum number deals to sample $P_{\max}$, minimum action probability to consider $p_{\min}$, maximum number of actions for search k

Output: Action after search

```
$H \leftarrow \text{ActionHistory}(h)$
$A \leftarrow$ top-$k$ actions from $\pi_\theta(\cdot|h)$ with probability larger than $p_{\min}$
$V(a) \leftarrow 0$ for $a \in A$
$R \leftarrow 0, P \leftarrow 0$
while $P < P_{\max}$ and $R < R_{\max}$ do
    Sample a deal $d$ from $B_\phi(h)$ consistent with $h$ and form $h'$ based on $d$
    $P \leftarrow P+1$
    for $a' \in H$ do
        with probability $1-\pi_\theta(a'|h')'$, skip to next deal
        $h' \leftarrow \mathcal{T}(h',a')$
    end for
    for $a \in A$ do
        $V(a) \leftarrow V(a) + \text{Rollout}(h,a,\pi_\theta)$
    end for
    $R \leftarrow R+1$
end while
if $R > R_{\min}$ then
    return $\arg\max_{a\in A} V(a)$
else
    return $\arg\max_{a} \pi_\theta(\cdot|h)$
end if
```

Algorithm 4 Rollout

Input: Game node h, candidate action a, policy network π_θ

Output: The evaluation of action a

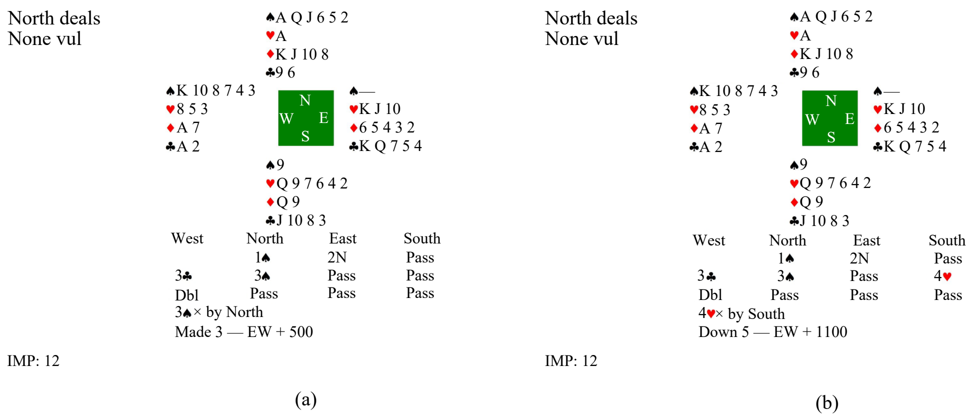
```
$h' \leftarrow \mathcal{T}(h,a)$
while $h'$ is not a terminal node do
    select greedy action $a' = \arg\max_{a} \pi_\theta(\cdot|h')$
    $h' \leftarrow \mathcal{T}(h',a')$
end while
return Reward for player making an action $a$ at $h$ computed based on the double dummy results and the contract reached at $h'$.
```

Appendix E
Case Study

We provide bidding cases from games played by our agents against WBridge5 for further analysis in this section.

Fig. 8 presents an illustrative collaborative bidding scenario, involving our RL agents positioned in the East and West seats. In this scenario, the East agent with its hand featuring 6♠ cards, makes an opening bid of 1♠. This decision not only aligns with the SAYC conventions but also matches the bid made by WBridge5 at the other table. Our RL agent, situated in the East position, effectively communicates and elevates the bid to a level 6 contract. Conversely, at the other table, WBridge5 opts to cease the contract at level 4. This decision, as it turns out, proves to be less advantageous, resulting in a noteworthy loss of 11 IMPs. This case exemplifies the strength of our RL agent in bid optimization through strategic communication, ultimately yielding a significant competitive advantage.

Fig. 9 provides an illustration of a competitive bidding scenario in which our agents occupy the North and South seats. After North makes a call of 3♠, our South agent, demonstrating its strategic prowess, elects to pass at this juncture. This decision is aimed at minimizing the potential penalties that might arise from pursuing a higher bid. On the other hand, at the other table, WBridge5 takes a different course of action by making a more aggressive bid of 4♡ with identical first calls. However, this choice results in a more substantial loss of scores for the WBridge5 partnership. In stark contrast, our agents exhibit their bidding finesse by securing a resounding



North deals
None vul

North: ♠A Q J 6 5 2 ♥A ♦K J 10 8 ♣9 6
West: ♠K 10 8 7 4 3 ♥8 5 3 ♦A 7 ♣A 2
East: ♠— ♥K J 10 ♦6 5 4 3 2 ♣K Q 7 5 4
South: ♠9 ♥Q 9 7 6 4 2 ♦Q 9 ♣J 10 8 3

West	North	East	South
	1♠	2N	Pass
3♣	3♠	Pass	Pass
Dbl	Pass	Pass	Pass

3♠× by North
Made 3 — EW + 500

IMP: 12

(a)

North deals
None vul

North: ♠A Q J 6 5 2 ♥A ♦K J 10 8 ♣9 6
West: ♠K 10 8 7 4 3 ♥8 5 3 ♦A 7 ♣A 2
East: ♠— ♥K J 10 ♦6 5 4 3 2 ♣K Q 7 5 4
South: ♠9 ♥Q 9 7 6 4 2 ♦Q 9 ♣J 10 8 3

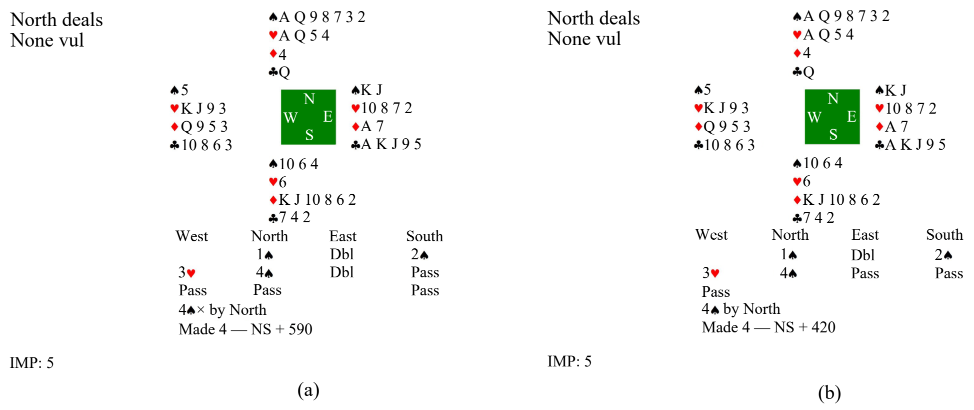
West	North	East	South
	1♠	2N	Pass
3♣	3♠	Pass	4♥
Dbl	Pass	Pass	Pass

4♥× by South
Down 5 — EW + 1100

IMP: 12

(b)

Fig. 9. A competitive bidding case played by RL agent ((a) The table where our RL agents sit at North and South; (b) The table where our RL agents sit at East and West).



North deals
None vul

North: ♠A Q 9 8 7 3 2 ♥A Q 5 4 ♦4 ♣Q
West: ♠5 ♥K J 9 3 ♦Q 9 5 3 ♣10 8 6 3
East: ♠K J ♥10 8 7 2 ♦A 7 ♣A K J 9 5
South: ♠10 6 4 ♥6 ♦K J 10 8 6 2 ♣7 4 2

West	North	East	South
	1♠	Dbl	2♠
3♥	4♠	Dbl	Pass
Pass	Pass		Pass

4♠× by North
Made 4 — NS + 590

IMP: 5

(a)

North deals
None vul

North: ♠A Q 9 8 7 3 2 ♥A Q 5 4 ♦4 ♣Q
West: ♠5 ♥K J 9 3 ♦Q 9 5 3 ♣10 8 6 3
East: ♠K J ♥10 8 7 2 ♦A 7 ♣A K J 9 5
South: ♠10 6 4 ♥6 ♦K J 10 8 6 2 ♣7 4 2

West	North	East	South
	1♠	Dbl	2♠
3♥	4♠	Pass	Pass
Pass			

4♠ by North
Made 4 — NS + 420

IMP: 5

(b)

Fig. 10. A case where the RL and RL-BMCS agents exhibit different performance ((a) The bidding history of RL agent against WBridge5; (b) The bidding history of RL-BMCS agent against WBridge5).

victory, ultimately outscoring their opponents by a remarkable 12 IMPs. This instance underscores the strategic strength and decision-making prowess of our agents in competitive bidding scenarios.

Fig. 10 provides an interesting case study where the RL agent and RL-BMCS agent exhibit distinct performances. In both instances, our agents are positioned in the West and East seats, and the initial calls made are identical. However, a pivotal divergence in their decision-making unfolds during the bidding phase. The RL-BMCS agent, demonstrating its strategic acumen, opts for a judicious pass call. This decision, notably, follows a meticulous search process aimed at assessing the probability of North and South successfully fulfilling the 4♠ contract. This shrewd maneuver leads to a substantial improvement in performance, amassing a noteworthy +5 IMPs advantage over the strategy of the RL agent.

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[3] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[4] J. Li, S. Koyamada, Q. Ye, G. Liu, C. Wang, R. Yang, L. Zhao, T. Qin, T.-Y. Liu, and H.-W. Hon, "Suphx: Mastering Mahjong with deep reinforcement learning," arXiv preprint arXiv: 2003.13590, 2020.

[5] 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.

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

[7] D. Zha, J. Xie, W. Ma, S. Zhang, X. Lian, X. Hu, and J. Liu, "Douzero: Mastering Doudizhu with self-play deep reinforcement learning," in *Proc. Int. Conf. Machine Learning*. PMLR, 2021, pp. 12333–12344.

[8] G. Yang, M. Liu, W. Hong, W. Zhang, F. Fang, G. Zeng, and Y. Lin, "Perfectdou: Dominating Doudizhu with perfect information distillation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 34 954–34 965, 2022.

[9] A. Lerer, H. Hu, J. Foerster, and N. Brown, "Improving policies via search in cooperative partially observable games," in *Proc. AAAI Conf. Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 7187–7194.

[10] H. Hu, A. Lerer, N. Brown, and J. Foerster, "Learned belief search: Efficiently improving policies in partially observable settings," arXiv preprint arXiv: 2106.09086, 2021.

[11] N. Brown, A. Bakhtin, A. Lerer, and Q. Gong, "Combining deep reinforcement learning and search for imperfect-information games," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17057–17069, 2020.

[12] M. Mazouchi, M. B. Naghibi-Sistani, and S. K. H. Sani, "A novel distributed optimal adaptive control algorithm for nonlinear multi-agent differential graphical games," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 1, pp. 331–341, 2018.

[13] J. Wang, Y. Hong, J. Wang, J. Xu, Y. Tang, Q.-L. Han, and J. Kurths, "Cooperative and competitive multi-agent systems: From optimization to games," *IEEE/CAA J. Autom. Sinica*, vol. 9, no. 5, pp. 763–783, 2022.

[14] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative multi-agent games," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 611–24 624, 2022.

[15] M. L. Ginsberg, "GIB: Steps toward an expert-level bridge-playing program," in *Proc. Int. Joint Conferences Artificial Intelligence*. Citeseer, 1999, pp. 584–593.

[16] H. Kuijf, W. Heemskerk, and M. Pattenier, "Jack bridge," [online]. Available: https://www.jackbridge.com, 2006.

[17] Y. Costel, "WBridge5," [online]. Available: http://www.wbridge5.com, 2014.

[18] A. Levy, "World computer-bridge championship," [online]. Available: https://bridgebotchampionship.com/, 2017.

[19] NukkAI, "Nook," [online]. Available: https://nukk.ai, 2022.

[20] E. Lockhart, N. Burch, N. Bard, S. Borgeaud, T. Eccles, L. Smaira, and R. Smith, "Human-agent cooperation in Bridge bidding," arXiv preprint arXiv: 2011.14124, 2020.

[21] V. Ventos, Y. Costel, O. Teytaud, and S. Thépaut Ventos, "Boosting a bridge artificial intelligence," in *Proc. IEEE 29th Int. Conf. Tools with Artificial Intelligence*, 2017, pp. 1280–1287.

[22] J. Rong, T. Qin, and B. An, "Competitive Bridge bidding with deep neural networks," in *Proc. 18th Int. Conf. Autonomous Agents and MultiAgent Systems*, 2019, pp. 16–24.

[23] Y. Tian, Q. Gong, and Y. Jiang, "Joint policy search for multi-agent collaboration with imperfect information," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19931–19942, 2020.

[24] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," arXiv preprint arXiv: 1606.08415, 2016.

[25] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, vol. 12, 1999.

[26] Q. Gong, Y. Jiang, and Y. Tian, "Simple is better: Training an end-to-end Contract Bridge bidding agent without human knowledge," [online]. Available: https://openreview.net/forum?id=SklViCEFPH, 2020.

[27] B. Haglund, "Double dummy solver," [online]. Available: https://github.com/dds-bridge/dds, 2018.

[28] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, *et al*., "Grandmaster level in StarCraft Ⅱ using multi -agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[29] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," arXiv preprint arXiv: 1511.05952, 2015.

[30] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Machine Learning*. PMLR, 2016, pp. 1928–1937.

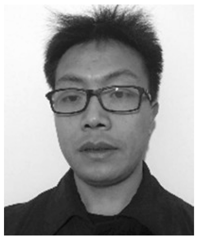
[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv: 1412.6980, 2014.

[32] P. Zhang, S. Shu, and M. Zhou, "An online fault detection model and strategies based on SVM-grid in clouds," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 2, pp. 445–456, 2018.

[33] X. Xie, P. Zhou, H. Li, Z. Lin, and S. Yan, "Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models," arXiv preprint arXiv: 2208.06677, 2022.

[34] M. Lanctot, E. Lockhart, J.-B. Lespiau, V. Zambaldi, S. Upadhyay, J. Pérolat, S. Srinivasan, F. Timbers, K. Tuyls, S. Omidshafiei *et al*., "Openspiel: A framework for reinforcement learning in games," arXiv preprint arXiv: 1908.09453, 2019.

Zizhang Qiu received the B.S. degree in electronic information science and technology from Southwest Jiaotong University in 2021. He is currently pursuing the master degree with the School of Information and Electronic Engineering, Zhejiang Gongshang University. His research interests include deep learning, reinforcement learning, and game theory.



Shouguang Wang (Senior Member, IEEE) received the B.S. degree in computer science from the Changsha University of Science and Technology in 2000, and the Ph.D. degree in electrical engineering from Zhejiang University in 2005. He joined Zhejiang Gongshang University in 2005, where he is currently a Professor with the School of Information and Electronic Engineering, the Director of the Discrete-Event Systems Group, and the Dean of System modeling and Control Research Institute, Zhejiang Gongshang University. His research interests include discrete event systems, Petri nets, automation, and AI.

Dan You (Member, IEEE) received the B.S. degree and the M.S. degree from the School of Information and Electronic Engineering, Zhejiang Gongshang University, in 2014 and 2017, respectively, and the Ph.D degree from the Department of Electrical and Electronic Engineering, University of Cagliari, Italy, in 2021. She is currently an Associate Research Professor with the School of Information and Electronic Engineering, Zhejiang Gongshang University. Her research interests include supervisory control of discrete event systems, fault prediction, and deadlock control, and siphon computation in Petri nets.

MengChu Zhou (Fellow, IEEE) Received the B.S. degree in control engineering from Nanjing University of Science and Technology in 1983, the M.S. degree in automatic control from Beijing Institute of Technology in 1986, and the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, USA in 1990. He joined the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, USA in 1990, and has been a Distinguished Professor since 2013. He is presently with Zhejiang Gongshang University. His research interests include intelligent automation, robotics, Petri nets, Internet of Things, edge/cloud computing, and big data analytics. He has over 1200 publications including 17 books, over 850 journal papers including over 650 IEEE Transactions papers, 31 patents and 32 book-chapters. His recently co-authored books include *Learning Automata and Their Applications to Intelligent Systems*, IEEE Press/Wiley, Hoboken, NJ, 2024 (with J. Zhang), *Device-Edge-Cloud Continuum Paradigms, Architectures and Applications*, Springer Nature, 2023 (with C. Savaglio, G. Fortino, and J. Ma) and *Sustainable Manufacturing Systems: An Energy Perspective*, IEEE Press/Wiley, Hoboken, Hoboken, NJ, 2022 (with L. Li). He is a recipient of Excellence in Research Prize and Medal from NJIT, Humboldt Research Award for US Senior Scientists from Alexander von Humboldt Foundation, and Franklin V. Taylor Memorial Award and the Norbert Wiener Award from IEEE Systems, Man, and Cybernetics Society, and Edison Patent Award from the Research & Development Council of New Jersey. He is a Life Member of Chinese Association for Science and Technology-USA and served as its President in 1999. He is Fellow of IEEE, International Federation of Automatic Control (IFAC), American Association for the Advancement of Science (AAAS), Chinese Association of Automation (CAA) and National Academy of Inventors (NAI).