Automatic Bridge Bidding Using Deep Reinforcement Learning

Chih-Kuan Yeh, Cheng-Yu Hsieh, Hsuan-Tien Lin *Member, IEEE*

***Abstract*—Bridge is among the zero-sum games for which artificial intelligence has not yet outperformed expert human players. The main difficulty lies in the bidding phase of bridge, which requires cooperative decision making with partial information. Existing artificial intelligence systems for bridge bidding rely on, and are thus restricted by, human-designed bidding systems or features. In this work, we propose a flexible and pioneering bridge bidding system, which can learn either with or without the aid of human domain knowledge. The system is based on a novel deep reinforcement learning model, which extracts sophisticated features and learns to bid automatically based on raw card data. The model includes an upper-confidence-bound algorithm and additional techniques to achieve a balance between exploration and exploitation. We further study how different pieces of human knowledge can be exploited to assist the model. Our experiments demonstrate the promising performance of our proposed model. In particular, the model can advance from having no knowledge on bidding to achieving a superior performance compared with a champion-winning computer bridge program that implements a human-designed bidding system. In addition, further synergies can be extracted by incorporating expert knowledge into the proposed model.**

***Index Terms*—Partial information games, bridge, deep reinforcement learning, Q-learning.**

I. Introduction

GAMES have always provided a challenging testbed for artificial intelligence (AI). Even for games with simple and well-defined rulesets, AI often must follow highly complex strategies to achieve victory. One line of research on AI for games focuses on full information games, including chess, Go, and Othello [1], whereas other research considers incomplete information games such as poker and bridge [2], [3], [4]. In both cases, traditional methods usually excel by embedding the knowledge of the best human players as computable strategies. However, researchers have recently shifted their focus to machine learning, allowing AI players to develop effective strategies automatically using data [1], [2], [4].

Bridge, which is a standard 52-card game that requires players to act both cooperatively and competitively, is one of the most appraised partial-information games for both humans and AI. The four players of the bridge game are commonly referred to as North, East, West, and South, and form two opposing teams (North-South and East-West). Each team aims to achieve the highest score in a zero-sum scenario.

When initiating this work, C.-K. Yeh was with the Department of Electrical Engineering and C.-Y. Hsieh was with the Department of Computer Science and Information Engineering at National Taiwan University. H.-T. Lin (htlin@csie.ntu.edu.tw) is with the Department of Computer Science and Information Engineering at National Taiwan University.

A single bridge game starts with a deal, followed by two phases: bidding and playing. A deal distributes 13 random cards to each player, and the cards are hidden from the other players—each player only sees partial information about the deal. In the bidding phase an auction is run to determine the declarer of the contract, where the contract affects the score that the declarer's team can achieve in the playing phase. The auction proceeds around the table in a clockwise manner, where each player chooses from one of the following actions: PASS, increasing the current value of the bid with respect to an ordered set of calls $\{1\clubsuit, 1\diamondsuit, 1\heartsuit, 1\spadesuit, 1NT, 2\clubsuit, ..., 7NT\}$, DOUBLING and REDOUBLING. The first two actions are general for deciding the contract, while the latter two are special, less-used actions, which modify the scoring function in the playing phase. The bidding sequence ends when three consecutive PASSes are placed, and the last bid becomes the final contract. The number occurring in the final contract (such as four in 4♠) plus six becomes the number of rounds that a team aims to win in the playing phase in order to satisfy the contract (commonly referred to as a "make"), and the symbol (such as ♠) reflects the trump suit in the playing phase.

There are 13 rounds in the playing phase of a bridge game, where each player shows one card from their hand and compares the values of the cards based on some rulesets, with the trump suit having some priority. The player with the highest-valued card among the four is the winner of the round. After the 13 rounds, the score of the declarer's team is calculated from a lookup table, based on the final contract and the number of winning rounds of the declarer's team, where making the contract results in a positive score for the declarer's team, and not making (failing) the contract results in a positive score for the opponent's team.

Bidding is an understandably difficult task, because of the incomplete-information setting. Given that each player can only see 13 out of 52 cards, it is impossible for a single player to infer the best contract for their team. Thus, each bid in the bidding phase must serve as a suggestion towards an optimal contract, an information exchange between team members, or both. That is, a good bidding strategy should strike a balance between exploration (exchanging information) and exploitation (deciding an optimal contract). Nevertheless, because the bid value must be monotonically increasing during the auction, the exchange of information is constrained to the extent that the bid cannot exceed the optimal contract. The constraint reduces the amount of exchangeable information. It is also possible that the two opposing teams may attempt to exchange information during the bidding phase, called bidding with competition, which blocks the other team's information-exchanging opportunities.

In real human bridge games, the best human players are often indistinguishable in terms of their professional competence in the playing phase. Thus, their competence in the bidding phase is the primary game-deciding factor. The abovementioned facts indicate the relative difficulty of the bidding phase over the playing phase for human players. This difficulty also holds true in the case of designing AI players. In the playing phase, it has been shown that AI players are competitive against professional human players. For example, in 1998 the GIB program finished in 12th place among 35 professional human players in a no-bidding bridge contest [5]. Nevertheless, in the bidding phase most existing AI players are based on replicating human-designed rules of bidding, commonly referred to as human bidding systems [6], [7], [8], [9]. This replication generally makes AI players less competitive than human players in the bidding phase, as explained below.

One of the main difficulties in replicating a human bidding system is the inevitable ambiguity of the bids. Human bidding systems are designed to have rules that cover different situations, but these rules can overlap. Therefore, based on the cards of one player and the other players' bids, conflicting suggestions could be arrived at from different rules, with every suggestion being a legitimate bid in the system. Human players are expected to resolve this ambiguity intelligently, and select an appropriate choice from the conflicting suggestions. In addition, professional human players devote a considerable amount of time to practicing together with team members, to reduce the ambiguity through mutual understanding. When AI players try to replicate human bidding systems, it is extremely challenging to reach the same level of mutual understanding that human players can achieve to resolve ambiguities, making AI players inferior in the bidding phase.

The ambiguity of bids arises primarily because human bidding systems need to be simple enough to be memorizable by human players. Thus, the rules within such systems are often also simple. On the other hand, if there were a bidding system for AI players rather than human players, the rules may not need to be so simple, and the ambiguity issue could be resolved to improve the performance of AI players in the bidding phase.

The aforementioned ideas have motivated some existing studies on enhancing AI for bridge bidding. Some studies begin by considering a human bidding system, and then resolve the ambiguity using various techniques. For instance, the combination of lookahead search with a human bidding system was studied by Gambäck et al. [10] and Ginsberg [5]. Amit and Markovitch [11] constructed a decision tree model with Monte Carlo sampling on top of a human bidding system to resolve the ambiguity of bids. DeLooze and Downey [12] generated examples from a human bidding system, and then used these examples as input for a self-organizing map for ambiguity resolution. In those approaches, human bidding systems play a central role in AI players' bidding strategies.

A more aggressive route to achieving bridge-bidding AI is to teach an AI about bidding without reference to a human bidding system. This was considered by Ho and Lin [2], who proposed a decision tree model along with a contextual bandit algorithm. The model exhibits the possibility to learn to bid directly, in a data-driven manner [2]. Nevertheless, the study is somewhat constrained by the decision tree model, which comes with a restriction of having at most five choices per bid (decision-tree branch). Moreover, because of the simple linear nodes in the decision tree, the model requires a more sophisticated feature representation of the cards. Ho and Lin [2] thus borrowed human knowledge on bidding by encoding the cards as human-designed features, such as the number of cards for each suit. The restrictions on bidding choices and feature representation limit the potential for building a data-driven bidding system for AI.

In all approaches discussed thus far, human-designed features are important, whether for the human bidding systems within the AI players [12], or for the AI bidding system being learned [2]. In [2], it was reported that raw-card features resulted in a considerably worse performance than human-designed features. Inspired by the recent success of deep learning in automatically constructing useful features from raw and abstract ones [1], we propose a novel framework that applies deep reinforcement learning for automatic bridge bidding, which contributes to the advancement of bridge-bidding AI in two aspects:

- learning data representation: Using deep neural networks for feature extraction, our proposed deep reinforcement learning model is the first that automatically learns the bidding system directly from raw data. Learning from raw data, without relying on human-designed bidding systems or human-designed features, unleashes the full power of machines for using all possible information. The promising performance of our proposed model showcases that learning a bidding system automatically with no human knowledge is possible.
- resolving bid ambiguity: As discussed previously, the main difficulty of bridge bidding is the ambiguity of bids. Using the proposed reinforcement learning framework, sophisticated bidding rules can be learned automatically to alleviate the ambiguity problem. The reinforcement learning framework arguably mimics the process of human players in establishing a mutual understanding by practicing together. However, in this case the framework does so "together" *with itself*. To the best of our knowledge, this is the first framework that achieves a promising mutual understanding in bridge bidding using only machines.

Moreover, we also demonstrate the flexibility of the proposed deep reinforcement learning framework by designing some add-ons that incorporate human knowledge into the learning process. Experiments demonstrate that the resulting human-knowledge-aware deep reinforcement model is able to not only enhance existing human bidding systems, but also achieve a better performance, indicating that synergies can be created by embedding domain knowledge into the proposed framework.

In summary, our proposed deep reinforcement learning framework enables the learning of complex rules of bidding using nonlinear functions on raw data, to avoid ambiguity of the bids and improve the bidding performance. In Section II, we formally establish the problem of bridge bidding as a learning problem. In Section III, we first introduce reinforcement learning and analyze the key issues in solving the bidding problem. Then, we propose a novel deep reinforcement learning framework based on a modification of the classic Q-learning algorithm, along with upper-confidence-bound algorithms for balancing between exploration and exploitation. We also introduce a modification of the Bellman's Equation, named the penetrative Bellman's Equation, to improve Q-learning. In addition, we present several add-ons for the proposed framework to incorporate human knowledge during training. Finally, in Section IV, we discuss our experiments, which demonstrate the promising performance of our proposed AI player based on the deep reinforcement learning framework. We demonstrate that the player's bidding performance compares favorably against state-of-the-art AI bidding systems [2] and a contemporary champion-winning bridge software, Wbridge5, which implements a human bidding system. Comparisons between models trained with and without expert knowledge are also studied, to verify the flexibility of our proposed framework.

A short version of the paper appeared in the 2016 European Conference on Artificial Intelligence [13]. The paper has since been enriched by a thorough round of polishing, a smoother introduction of the connection between deep reinforcement learning and the bridge bidding task in Section II, a broader discussion of the possible add-ons for the proposed model for incorporating human knowledge in Section III, and a careful comparison of the models with and without human knowledge in Section IV.

II. Problem Setup

We shall first illustrate the bridge bidding problem, and then connect it to reinforcement learning.

A. Bidding Problem

The general bidding problem can be divided into two subproblems, namely bidding without competition, and bidding with competition [2], both of which contain significant numbers of deals in actual bridge games. Bidding without competition assumes that the opponent's team always calls $PASS$ during bidding, and hence information exchange is not blocked. Bidding with competition means that both teams want to bid. In this study, we focus on the subproblem of bidding without competition, as in existing studies [2], [11], [12].

Most human bidding systems are designed from "general cases" of bidding without competition, and treat the possible competitions as "special cases" to be handled. The design is because it is readily challenging to achieve a sufficient level of mutual understanding between partners to play against $PASS$-only opponents under the situation of bidding without competition, and the situation happens rather frequently in real-world games. Thus, the subproblem of bidding without competition is an important one on its own, and allows us to evaluate the basic bidding level between partners. In fact, the subproblem has been a longstanding benchmark, called the bidding challenge, for testing computerized bidding systems [7], [8], [9] as well as evaluating the performance of human bridge players during practice sessions.

For simplicity, we assume that the bidding team is composed of two players, Player 1 and Player 2, who sit at the North-South positions, and their opponents always bid PASS. The two players are generally called partners in the bidding process. Without loss of generality, Player 1 is assumed to take the first and the other odd turns to bid, while Player 2 is assumed to bid in the even turns.

We use $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ to denote the cards of Player 1 and Player 2, respectively, and $\boldsymbol{b}$ to denote the bidding history. The element $\boldsymbol{x}_i[k]$ of the boolean array $\boldsymbol{x}_i$ indicates whether Player i holds the k^{th} card in the ordered set $\{\spadesuit 2, \spadesuit 3, ...\spadesuit A, \heartsuit 2, \ldots, \heartsuit A, \diamondsuit 2, \ldots, \diamondsuit A, \clubsuit 2, \ldots, \clubsuit A\}$. Notice that the bidding history $\boldsymbol{b}$ updates as the bidding process proceeds. Thus, we explicitly denote by $\boldsymbol{b}^{(t)}$ the bidding history up to the first t turns. The element $\boldsymbol{b}^{(t)}[j]$ of the boolean array $\boldsymbol{b}^{(t)}$ indicates whether Player 1 or Player 2 has made the j^{th} bid in the ordered set $\beta = \{PASS, 1\clubsuit, 1\diamondsuit, \ldots, 7NT\}$. As stated in bridge rules, a new bid must be higher than all previous bids. Henceforth, it is sufficient to infer who made which bid given $\boldsymbol{b}^{(t)}$. For example, if $\boldsymbol{b}^{(5)}$ contains $\{PASS, 1\spadesuit, 1NT, 3\spadesuit\}$, then Player 1 must have bid $\{PASS, 1NT\}$ and Player 2 must have bid $\{1\spadesuit, 3\spadesuit\}$.[1]

To incorporate the partially observable nature of bridge bidding, we define $\boldsymbol{s}^{(t)}$, the state at turn t, as containing two components:

- $\boldsymbol{x}^{(t)}$: the hand of the corresponding player in turn t, where $\boldsymbol{x}^{(t)} = \boldsymbol{x}_1$ for odd t and $\boldsymbol{x}^{(t)} = \boldsymbol{x}_2$ for even t.
- $\boldsymbol{b}^{(t)}$: the bidding history up to turn t.

By the definition, each player is restricted to accessing their own hand when deciding the next bid. The goal is to find a strategy $G(\boldsymbol{s}^{(t)}) = a^{(t)}$, where the action (bid) $a^{(t)}$ is among the ordered set $\beta = \{PASS, 1\clubsuit, 1\diamondsuit, \ldots, 7NT\}$. Except for the case of $PASS$, G must satisfy the constraint $a^{(t)} > a^{(t-1)}$ to satisfy the rules of bridge. We relax this constraint slightly and allow G to use $a^{(t)} = a^{(t-1)}$ to represent agreement with the previous action, which is effectively the same as calling $PASS$ in an actual bridge game, but technically gives the machine a slightly bigger action space to express the intent of terminating the bidding procedure. For any given strategy G, the array $\boldsymbol{b}^{(t)}$ will be updated by the bid $a^{(t)}$ of the strategy in each bidding turn.

The data used to learn a good bidding strategy G are generated as follows. Each instance within the data is of the form $(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{c})$, where $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ represents a particular deal to the North-South players, and $\boldsymbol{c}$ carries the information regarding the goodness (more specifically the relative penalty, as we shall explain next) of each possible contract with respect to $(\boldsymbol{x}_1, \boldsymbol{x}_2)$. It would be extremely time consuming for humans

[1] The use of $\boldsymbol{b}$ to represent the bidding history was specially designed for bidding without competition out of simplicity. More sophisticated representation is needed for future studies of bidding with competition.

or machines to actually play out each possible contract to learn the goodness score. Thus, we calculate the score of each contract without actually carrying out the playing phase. In particular, we use double dummy analysis, as in previous studies [2], to estimate the score for each possible contract.

Double dummy analysis is a technique that attempts to compute the number of winning rounds of each team in the playing phase with perfect information and an optimal playing strategy, and is generally considered to be a solved problem in the field of bridge AI. Although the analysis is performed with the optimistic assumption of perfect information, it is adopted here for two reasons. First, the results are independent from the bidding stage, making it possible to generate a large amount of data efficiently. Second, double dummy analysis is known to achieve a considerable accuracy when compared with actual games played by professional human players.

Double dummy analysis allows the goodness of each possible contract to be calculated with respect to one particular deal of hands. Nevertheless, even with the North-South hands fixed, the distribution of East-West hands may affect the goodness of each contract. Human players generally bid to maximize the expected score with respect to the opposing team's hands. We approximate the expected score by dealing the remaining cards to the East and West players five times for any given North-South hands $(\boldsymbol{x}_1, \boldsymbol{x}_2)$, performing a double dummy analysis for each deal, and averaging over the five analysis results to obtain the final score of each possible contract. This averaging reduces the "noise" resulting from luckiness within the data, and provides the learning algorithm with more accurate information to learn from. After obtaining the final score for each possible contract, we store the absolute difference between the final score and the highest final score in a cost vector $\boldsymbol{c}$, where $\boldsymbol{c}[j]$ indicates the penalty of reaching a final contract j.

We now formally define our learning problem as follows. Given a data set $D = \{(\boldsymbol{x}_{1n}, \boldsymbol{x}_{2n}, \boldsymbol{c}_n)\}_{n=1}^{N}$, where N is the number of instances, we aim to learn a bidding strategy G. For each $(\boldsymbol{x}_1, \boldsymbol{x}_2)$, the strategy G is iteratively fed the current state $\boldsymbol{s}^{(t)} = (\boldsymbol{x}^{(t)}, \boldsymbol{b}^{(t)})$ until it calls $PASS$ or the same bid. We denote the state for which G calls the final bid (contract) by $\hat{\boldsymbol{s}}$. The cost of the contract, namely $\boldsymbol{c}[G(\hat{\boldsymbol{s}})]$, is then used to evaluate G. Our goal is to minimize the expected test cost of G.

B. Reinforcement Learning

Next, we shall explain how the goal of learning the bidding strategy G is similar to the goal of reinforcement learning. A reinforcement learning problem is generally represented by a set of possible environment states, and a set of actions that can be taken on each state. In each iteration t, the learner takes an action a in a state $\boldsymbol{s}$. The action makes the environment state change from $\boldsymbol{s}$ to some $\boldsymbol{s}'$, and results in some reward r_{t+1} for the learner as feedback. The goal of reinforcement learning is to obtain the maximum (discounted) cumulative reward $\sum_{t=0}^{\infty} \gamma^t r_{t+1}$, where the discount $0 < \gamma \leq 1$ represents the intent of obtaining the reward as early as possible.

Q-learning [14] is a technique of reinforcement learning that evaluates the goodness, denoted as $Q(\boldsymbol{s}, a)$, of taking each action a in some state $\boldsymbol{s}$. The optimal action-value function $Q^*(\boldsymbol{s}, a)$ is defined as the maximum expected return that is achievable for any strategy after performing an action a in a state $\boldsymbol{s}$. Similarly, $Q^*(\boldsymbol{s}, \boldsymbol{a})$ is the vector in which each value is obtained as the maximum expected return that is achievable with any strategy after performing the corresponding action in the vector $\boldsymbol{a}$ and state $\boldsymbol{s}$.

The optimal action-value function obeys an important equation, known as the Bellman equation. The Bellman equation assumes that given the current state $\boldsymbol{s}$, the resulting state $\boldsymbol{s}'$ after taking action a, and all possible actions a' in $\boldsymbol{s}'$, the optimal value of $Q^*(\boldsymbol{s}, a)$ is the expected sum of the instant reward r and the total rewards after the best action among a' is executed. Formally, we have that

$$Q^*(\boldsymbol{s}, a) = \mathbb{E}_{\boldsymbol{s}'}[r + \gamma \max_{a'} Q^*(\boldsymbol{s}', a') | \boldsymbol{s}, a] \quad (1)$$

The general idea behind reinforcement learning is to obtain an estimate of the optimal Q-function by continuously modifying it based on feedback from actions. In particular, when taking an action a in the state $\boldsymbol{s}$ to move to the state $\boldsymbol{s}'$ and obtain a reward r, a fixed-point style update can be performed on the entry $(\boldsymbol{s}, a)$ of the Q-function table as follows:

$$Q(\boldsymbol{s}, a) \leftarrow \alpha Q(\boldsymbol{s}, a) + (1 - \alpha)[r + \gamma \max_{a'} Q(\boldsymbol{s}', a') | \boldsymbol{s}, a] \quad (2)$$

where α is the learning rate. The updating algorithm, which is called the value iteration algorithm, provably converges to the optimal Q-value after exploring the space of $(\boldsymbol{s}, a)$ [15]. Nevertheless, Q-learning itself does not specify how to actually choose the actions and explore the space, as it is used only to construct a value function.

The value iteration algorithm works when the state-action space $(\boldsymbol{s}, a)$ is of finite size. For more complicated problems, a model-based Q-function can be employed, such as a deep neural network [16], [17]. In this case, a loss function is often introduced to measure the difference between the current and newly experienced Q-values. When taking an action a in the state $\boldsymbol{s}$ to move to the state $\boldsymbol{s}'$ and obtain a reward r, a Q-network can be trained by minimizing a squared-error loss function with respect to the network parameter θ:

$$L(\theta) = \Big(Q(\boldsymbol{s}, a; \theta) - [r + \gamma \max_{a'} Q(\boldsymbol{s}', a'; \theta) | \boldsymbol{s}, a]\Big)^2 . \quad (3)$$

One typical method of minimizing the above loss function is to apply stochastic gradient descent on each new example of $(\boldsymbol{s}, a, \boldsymbol{s}', r)$ obtained during the reinforcement learning process.

Using the above technique of (deep) Q-learning, we tackle the bridge bidding problem by reducing the task of choosing a bid a under the state $\boldsymbol{s}$ to the task of choosing an action a under the state $\boldsymbol{s}$ (hence our overloading of the same notations). Two challenges remain to be solved. First, how can we connect the cost of the final bid $\boldsymbol{c}$ to the rewards r obtained *during* the choice of each bid? Second, how can we properly explore the state-action space $(\boldsymbol{s}, a)$ to improve the reinforcement learning performance? We will tackle these challenges in the next section.

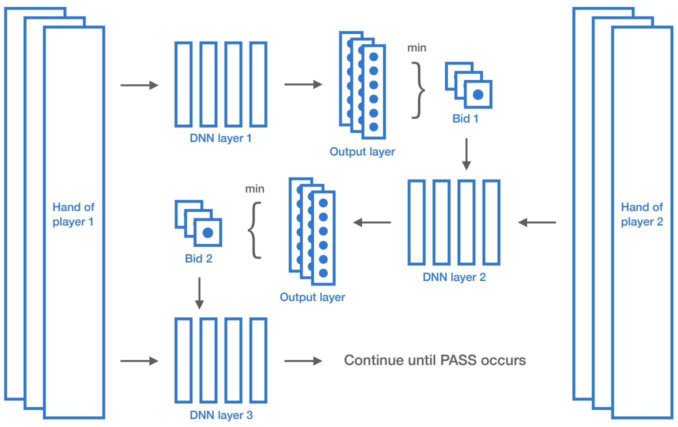


Fig. 1: The structure of our bridge-bidding deep reinforcement learning framework.

Algorithm 1: Proposed Learning Algorithm

```
Input: Data = $\{(\boldsymbol{x_{1n}},\boldsymbol{x_{2n}},\boldsymbol{c_n})\}$ for $n = 1,\dots,N$
Algorithm P to determine the cost of action $a$
Algorithm E to determine exploration and exploitation strategy
Output: A bidding strategy G based on the learned $\boldsymbol{\theta_i}$.
Initialize the action-value function $Q_j$ with random
  weights for $j = 1,\dots,l$
repeat
    Randomly select a data instance $(\boldsymbol{x_{1n}},\boldsymbol{x_{2n}},\boldsymbol{c_n})$
    for turn t = 1 to l do
        Initialize cost array $\boldsymbol{c}(\boldsymbol{a^{(t)}})$
        for all possible action $a^{(t)}$ do
            Determine the cost of action $a^{(t)}$ by P
            Record resulting cost in $\boldsymbol{c}(\boldsymbol{a^{(t)}})$
        Save $(S^{(t)},\boldsymbol{c}(\boldsymbol{a^{(t)}}))$ in Database D
        Select action $a^{(t)}$by the highest estimated reward
          with exploration by E
        if $a^{(t)} == PASS$ then
            Break
        Update $\boldsymbol{b^{(t+1)}}$ by action $a^{(t)}$
        Set $\boldsymbol{s}^{(t+1)} = (\boldsymbol{x^{(t+1)}},\boldsymbol{b}^{(t+1)})$
    for turn t = 1 to l do
        Sample random mini-batch of $(S^{(t)},\boldsymbol{c}(\boldsymbol{a^{(t)}}))$
          from D
        Perform a gradient descent step on
          $[(1-\boldsymbol{c}(\boldsymbol{a^{(t)}}))-Q(\boldsymbol{s}^{(t)},\boldsymbol{a^{(t)}};\theta)]^2$
until enough training iterations by early stopping
```

III. Model

We begin the introduction of our proposed framework by connecting the intermediate rewards with the final costs. Then, we discuss the design of our exploration technique. Finally, we illustrate the possibility of incorporating human knowledge into the framework.

A. Proposed Framework

The problem of learning a new bidding system can be complicated. It is considerably difficult to infer the full state from the bidding sequence alone, because each call can either suggest an optional contract or serve as an information exchange between partners. A different intention would lead to different inferences in the same situation, and thus partners often maintain a list of agreements regarding the meanings of their bids, widely known as a bidding system. Interestingly, the same exact bid may be considered to be good in one bidding system and bad in another. Therefore, while learning a new bidding system, players need to modify their interpretations of bids alongside with partners.

The bidding problem without competition can be viewed as a multi-agent game such that each bidder is seen as a different player at different stages of bidding. In this case, each player has a unique sequence number: players with an odd sequence number can share the same 13-card information, while players with even sequence number share another set of 13-card information. Each player knows the bidding result of all the players before her/him. The game stops when $PASS$ is bid by a player with a sequence number greater than one, or when an indication of PASS (two consecutive identical bids) is presented. Thus, we are able to separate the decision process of each layer by "training" a different Q-function for each layer of bidding. The algorithm is defined and illustrated in Algorithm 1.

In traditional Q-learning, the cost of each bid is updated using the Bellman equation, as in Equation 1. Moreover, the exploration behavior is often demonstrated by an ϵ-greedy strategy, which follows the greedy strategy with probability $1-\epsilon$ and selects a random action with probability ϵ. This forms the baseline algorithm.

While Bellman's equation is a necessary condition for optimality, the convergence time for each Q-function is rather long. Moreover, it has been shown that Q-learning performs considerably poorly in some stochastic environments, because of the overestimation of action values. In the problem of bridge bidding, this being a partial-information cooperative game, the overestimation of action values becomes a significant problem. Overestimation during the bridge bidding happens because of the fact that $\boldsymbol{s}^{(t)}$ and $\boldsymbol{s}^{(t+1)}$ observe the hands of different players (they observe the hands of Player 1 and Player 2, respectively), making the optimal Q-value for $\boldsymbol{s}^{(t+1)}$ almost impossible to estimate from $\boldsymbol{s}^{(t)}$. The partial information nature of the game thus raises a convergence issue for Q-learning based on the conventional Bellman equation.

To overcome this challenge, we propose the penetrative Bellman equation, which involves a one-step Monte Carlo sampling to obtain an estimate for the final score as the Q-learning target. By simulating the game, we can update the Q-value by the final score of the game outcome, instead of the intermediate Q-value estimated by observing the partner's hand. The mathematical formulation of the penetrative Bellman's equation is as follows. Starting from the Bellman's equation, we define the instant reward as zero and $\gamma = 1$, to obtain

$$\begin{aligned} Q^{(i)*}(\boldsymbol{s}, a) &= \max_{a^{(1)}}[Q^{(i+1)*}(\boldsymbol{s}^{(1)}, a^{(1)})|\boldsymbol{s}, a] \\ &= Q^{(i+1)*}(\boldsymbol{s}^{(1)}, a^{*(1)}|\boldsymbol{s}, a), \end{aligned} \tag{4}$$

where $a^{*(1)}$ is the best possible action from $\boldsymbol{s}^{(t)}$. We similarly define $a^{*(t)}$ to be the best possible action determined by the Q-value at state $\boldsymbol{s}^{(t)}$, where $\boldsymbol{s}^{(t)}$ is translated from $\boldsymbol{s}^{(t-1)}$ following the action $a^{(t-1)}$, and $(\boldsymbol{s}^{(0)}, a^{(0)}) = (\boldsymbol{s}, a)$. We could then further apply Bellman's equation to $Q^{(i+1)*}(\boldsymbol{s}^{(1)}, a^{*(1)})$ as:

$$Q^{(i+1)*}(\boldsymbol{s}^{(1)}, a^{*(1)}) = \max_{a^{(2)}}[Q^{(i+2)*}(\boldsymbol{s}^{(2)}, a^{(2)})|\boldsymbol{s}^{(1)}, a^{*(1)}] \tag{5}$$

while

$$\max_{a^{(2)}}[Q^{(i+2)*}(\boldsymbol{s}^{(2)}, a^{(2)})|\boldsymbol{s}^{(1)}, a^{*(1)}] = Q^{(i+2)*}(\boldsymbol{s}^{(2)}, a^{*(2)}|\boldsymbol{s}^{(1)}, a^{*(1)}) \tag{6}$$

By combining equation 4, equation 5, and equation 6, we have

$$Q^{(i)*}(\boldsymbol{s}, a) = Q^{(i+2)*}(\boldsymbol{s}^{(2)}, a^{*(2)}|\boldsymbol{s}, a) \tag{7}$$

where $\boldsymbol{s}^{(2)}, a^{*(2)}, \boldsymbol{s}^{(1)}, a^{*(1)}, \boldsymbol{s}$, and a satisfy equation 4 and equation 5. By recursively applying Bellman's equation, the process stops when $a^{*(t)}$ is the final bid of the game, and thus the cost of the game can be decided by the precalculated $\boldsymbol{c}(a^{*(t)})$. That is, the target Q-value for $(\boldsymbol{s}, a)$ is computed as

$$Q^{(i)*}(\boldsymbol{s}, a) = Q^{(i+t)*}(\boldsymbol{s}^{(t)}, a^{*(t)}|\boldsymbol{s}, a) = \boldsymbol{c}(a^{*(t)}) \tag{8}$$

Typically, there are fewer than six bids in a game of bridge bidding between the two teammates, and therefore the penetrative Bellman's equation is fairly efficient compared with the original variant, while the cost of each action can be more reliably estimated by resolving the different states observed by consecutive time steps. As discussed later in the experimental section, this leads to better performances.

Because bridge bidding is a multi-agent cooperative game, the traditional ϵ-greedy algorithm would be detrimental to communication between partners. Note that the main objective in bridge bidding, other than to find the best possible contract, is to convey some information to one's partner. However, randomly bidding any contract in lieu of a certain possibility would make it difficult for the partner of the bidder to understand the current bid. This will result in poor communication and a slower convergence, which has more disadvantages than advantages.

Further, exploration is one of the key elements for reaching the optimum in reinforcement learning. Without exploration, reinforcement learning will likely be confined to some local optimum because the value of some actions will never be explored. Various studies have investigated the problem of balancing exploration with exploitation. Previous research on exploration in reinforcement learning proposes the use of a sampling technique such as "Thompson sampling" to enhance the exploration performance [18], [19].

However, most related approaches cannot deal with one of the key differences between bridge bidding and traditional reinforcement learning: communication with the partner. One of the difficulties of learning a good bidding strategy is the complexity in exploring the value of an action. In games such as chess or Go, one may learn that a move is recommended by evaluating the state through playing afterwards. However, in the game of bridge, a bid is only good if one's partner understands the bid and is able to react accordingly. Even in the exploration phase, bids need to be consistent with the partner's knowledge. The standard ϵ-greedy exploration scheme in deep Q-network (DQN) which samples all bids with equal probability, will deteriorate the communication understanding between partners. The uniform randomness arguably introduces too much noise in the communication process, imposing more harm than good (exploration) for the learning algorithm, as we shall demonstrate in the experimental section.

Moreover, considering information theory, the exchange of information works best when the use of each bid is distributed equally. Therefore, we design an exploration scheme using a bandit algorithm. The bandit problem has been a popular research topic in the field of machine learning [20], [21], [22], [23]. In the contextual bandit problem, we would like to earn the maximum total rewards within finite attempts by pulling a bandit machine from M given machines in a dynamic environment with context. The key is to balance exploration and exploitation. Upper-confidence-bound (UCB) algorithms [21] are some of the most popular contextual bandit algorithms. These algorithms use the uncertainty term to achieve a balance. For the bridge-bidding problem, we choose to use UCB1 [20], for its simplicity in connecting with deep neural networks and its good performance in previous studies.

We now relate the bridge bidding problem to the contextual bandit problem. We assume that each possible bid is a bandit machine, with the context being the cards in one's hand and previous bids. The reward for each bid is calculated using the penetrative Bellman's equation, which relates to the final cost vector and future strategy. Nevertheless, there may be uncertainty in the terms of the action-value, especially for bids that are rarely used. Contextual bandit can be applied to achieve a balance between using the best action inferred by the Q-function and exploring bids that occur less frequently. The neural network of the Q-function serves as a non-linear version of the reward, the $W^T X$ term in UCB1. Therefore, we formally define algorithm E using UCB1 by selecting $a^{(t)} = \max_{a^{(t)}}[Q(\boldsymbol{s}^{(t)}, a^{(t)}; \theta) + \alpha\sqrt{\frac{2\ln T}{T_a}}]$, where T is the total number of examples used to learn the entirety of Q, and T_a is the number of examples such that the action a (bid a) has been selected. The benefit of using UCB1 for sampling is that the algorithm is effectively balanced between exploration and exploitation, exploring the less used bids while choosing the bid with lower estimated costs in general. Therefore, the exploration algorithm is able to explore without seriously hindering the communication between partners, which is especially critical in the case of cooperative multi-agent games. The final algorithm is presented in Algorithm 1.

B. Preprocessing and Model Architecture

The features of training bridge-bidding AIs in previous studies include bridge-specific features invented by humans, such as high-card points[2]. Somehow the best form of high-

[2]High-card points - total points for the picture cards: A=4, K=3, Q=2, J=1.

card points is debatable since 10s and 9s may well play an important role in certain hands. Because our proposed deep reinforcement learning model is able to conduct automatic feature extraction, we propose using 52-dimension raw hand data as our features. The potential advantage of the feature representation is to not limit the machines by human heuristics; the potential disadvantage is to risk of overfitting to raw features. We shall demonstrate that the representation is promising—a well-performing bridge-bidding system can be designed without human bridge knowledge.

There are several possible approaches to designing a Q-function using a neural network. One approach uses the bidding history and actions as inputs to the neural network, while another involves listing the costs of all possible outputs, only with the state as input. The drawback of the former is that the computational cost will increase linearly with the number of possible actions. Thus, we choose the latter approach, and therefore it can be stated that the output of the Q-function corresponds to a predicted cost vector of all possible bids. We denote the action vector by $\boldsymbol{a}$ and the true cost vector of all possible bids by $\boldsymbol{c}(\boldsymbol{a})$. The gradient descent update of the Q-function can be performed on $(\boldsymbol{c}(\boldsymbol{a}^{(t)}) - Q(\boldsymbol{s}^{(t)}, \boldsymbol{a}; \theta))^2$. Notably, there may be actions that are illegal in certain states because they violate the rule of bridge. We set the cost of such actions to an extremely high value, so that the rules of bridge can be learned by the Q-function explicitly.

We now describe the architecture of the Q-function of the bridge-bidding problem. We initialize l separate Q-functions, where l is the total number of bids. For the first Q-function, the input is the 52-dimension raw data of Player 1's hand using one-hot encoding, followed by three layers of fully-connected layers with 128 neurons each. We obtain a 36-dimension output for the cost of each bid. Compared with the first Q-function, for the other Q-functions, there are an extra 36 dimensions describing the bidding histories of the both players, where the 36 dimensions represent $\{PASS, 1\clubsuit, 1\diamondsuit, \ldots, 7NT\}$. Specifically, in the bridge-bidding scenario, a $PASS$ may lead to different final contracts depending on the bidding history, making the Q-value of $PASS$ difficult to learn. To facilitate the learning process, we encode a $PASS$ bid as repeating the previous bid by the partner, as an alternative representation of the action. In the modified representation, if the same bids occurs twice, this is equivalent to the latter bid being a $PASS$. This alternative representation is one of the key elements in the success of Q-learning for bridge bidding. The bids that have been placed by any of the two players have a value of one, while others have the value zero. The final structure of our learning framework is illustrated in Figure 1.

C. Incorporating the Opening Bid of Human Bidding Systems

Thus far, this paper has focused on learning the bidding system from scratch, without the aid of any human bridge knowledge. In the following subsections, we demonstrate the flexibility of the proposed model by studying different possibilities for incorporating human knowledge into the learning process. As an initial attempt to combine the deep reinforcement learning algorithm with human knowledge, we hope that synergies can be extracted to benefit both existing human bidding systems and our learning algorithm.

In every human-designed bidding system, there is a specific set of rules to follow at every stage in the bidding phase. Among these rules, the opening table, which defines what to bid for the opener, serves as an important signature for a human bidding system to distinguish it from others. Given the unique roles that opening bids play, we seek ways to combine human-designed opening bids with our deep reinforcement learning algorithm as a first step towards embedding human knowledge into our learning algorithm.

1) Hard opening with human bidding systems: Under the proposed framework in Algorithm 1, a natural approach to combining human-designed opening bids with our deep reinforcement learning algorithm is to fix the first bid using a written open table of the human bidding system of interest. That is, we force the learning algorithm to learn all the bids after the fixed opening bid. Nevertheless, it is known from [13] that such a strong condition on the opening bid may result in limiting the potential power of computer bridge bidding. Therefore, we propose a soft alternative to this approach below.

2) Soft opening with human bidding systems: As an alternative to strongly forcing the first bid to exactly follow the open table of some human bidding system, a more gentle manner of leveraging human knowledge is to assign a certain probability for our learning algorithm to select the opening bid according to the prescribed rules of human bidding systems. In other words, at the first bid, the learning algorithm has a probability of ϵ_h to directly choose an action suggested by some human bidding system, and a probability of $1 - \epsilon_h$ to follow the Q-learning strategy learned on its own. By introducing ϵ_h, the reinforcement learning model can perform exploration based on some prior human knowledge. Thus, we refer to this approach as ϵ-human. Using this soft version, we are able to control how strictly we want to enforce human knowledge on our algorithm through ϵ_h. We note that this is equivalent to the above mentioned hard version when $\epsilon_h = 1$, and equivalent to not considering any human-designed bidding system when $\epsilon_h = 0$.

D. Training with Human-Crafted Hand Features

We now explore an alternative possibility for our learning algorithm to take advantage of existing human knowledge in bridge. To declare an optimal contract, it is undoubtedly important for the partners to be able to exchange information and learn each other's hands during the bidding phase. As a result, [13] proposed to add a representation of the partner's hand in the output layer while training the deep reinforcement algorithm, with the hope that these extra features can guide the bidding algorithm to gain a better understanding of the partner's bid.

However, there is a huge variety of features that could be used to represent the hand, ranging from lower-level features such as the presence or absence of specific cards to higher-level human-crafted features such as the point-count system, suit distribution, or even more complex quality assessment features. Henceforth, we further generalize the original idea in [13] and attempt to utilize different levels of hand features to aid the learning algorithm. That is, we start by adding lower-level hand features to the more sophisticated features, and see how the learning algorithm reacts to different types of auxiliary features. In fact, this idea is aligned with the recently introduced concept of the auxiliary task [24], where the goal of the agent is to predict not only what action to perform, but also how the action affects the future state. As an analogy, predicting the partner's hand in bridge bidding can then be viewed as an auxiliary task. We validate the benefits of such auxiliary tasks in the following section.

IV. Experiment

To validate the proposed method, we compare our model with a baseline model, a state-of-the-art model, and a well-known computer bridge software, Wbridge5 [25], which has won the computer bridge championship for the last several years. We randomly generate a dataset of 140,000 instances (pairs of hand distributions) to be used in our experiments. We use 100,000 instances for training, and split the other 40,000 instances evenly for validation and testing. We also compare the sparse binary features for representing the existence of each card to the condensed features using a second-order extension employed in previous studies [2].

Note that Wbridge5 is originally designed for the problem of bidding with competition. To obtain its experimental results for bidding without competition, we follow an earlier benchmark [2] to set $PASS$ as the opponents' bids. Admittedly, the comparison is not fully fair to Wbridge5, as its bidding system readily contains rooms for potential competition. We intend to view Wbridge5 as a (strong) baseline to evaluate whether our proposed model is promising in the constrained setting of bidding without competition, with the hope that the positive results can inspire future studies and comparisons under the full setting of bidding with competition.

We set the cost vectors $\boldsymbol{c}(\boldsymbol{a})$ using International Match Points, which consists of an integer between 0 and 24 that is commonly used to compare the relative performances of two teams in most bridge games. The cost vector is obtained by subtracting the cost of action array from the best possible bid, followed by a normalization step; namely, $\boldsymbol{c}(\boldsymbol{a}) = [\boldsymbol{c}(\boldsymbol{a})' - min_a \boldsymbol{c}(a)']/25$. The division by 25 ensures that the cost is scaled to between 0 and 1 to facilitate training stability, while we report the non-scaled values in the table to reflect the original difference of International Match Points. Here, $\boldsymbol{c}(\boldsymbol{a})'$ denotes the origin cost of each action calculated by the double dummy analysis[3]. The cost can be transformed to the reward using $R(\boldsymbol{a}) = 1 - \boldsymbol{c}(\boldsymbol{a})$. We set the cost of a rule-violating bid to 1.2, thus letting the bidding system learn the bidding rules implicitly. Moreover, the bids in the testing phase are chosen from legal bids.

For deep neural networks, RMSprop is used to speed up the convergence time. In the following experiments, we fix the parameters related to the deep neural network. The following parameters were used in the experiments for the fully-connected deep neural network: decay = 0.98, momentum = 0.82, step rate = 0.83, batch size = 50, and η = 0.05. These parameters remain unchanged during our experiments, because the focus of our study is not on deep neural network parameters. We use early-stopping from the validation result to determine the number of epochs after which to end the training.

[3]One technical detail is that $\boldsymbol{c}$ is generated by assuming that the player who can win more tricks in the contract is the declarer.

A. Exploration Method

We compare the two exploration methods ϵ-greedy exploration and UCB1 for the exploration algorithm E in Algorithm 1. The parameter in ϵ-greedy exploration specifies choosing a random action with probability ϵ and following the best action given the Q-function otherwise. The parameter in UCB1 exploration specifies selecting $a^{(t)} = max_{a^{(t)}}[Q(\boldsymbol{s}^{(t)}, a^{(t)}; \theta) + \alpha\sqrt{\frac{2\ln T}{T_a}}]$, where T is the number of total examples used to learn the complete Q-function, and T_a is the number of examples in which action a (bid a) is selected. We perform experiments on $\epsilon \in \{0.001, 0.005, 0.01, 0.05, 0.1\}$ and $\alpha \in \{0.05, 0.1, 0.2\}$. The results are listed in Table I, where *no exploration* represents the outcome in which no exploring methods are used.

We can see from Table I that UCB1 exploration generally outperforms that using the ϵ-greedy approach, showing that UCB1 exploration fits the model well. The ϵ-greedy exploration method performs even worse than in the case of no exploration for $\epsilon \geq 0.05$, which is arguably because of the enhanced ambiguity of random exploration. It is worth noting that the no-exploration method does include some sense of exploration in the deep neural network structure itself, because all the possible actions are updated in certain Q-functions, and thus they also contain the actions that are not likely to be chosen. However, deep exploration, such as in the case of UCB1, can further improve the result, as shown in Table I. The best parameter for UCB1 exploration is α = 0.1 for layers $\geq$ 3 and α = 0.05 for layers = 2. We have conducted a Student's t-test at 95% confidence level between the best parameter and the second best parameter for each method/layer combination, and the results confirmed that the differences are statistically significant. These parameter values will be adopted in the experiments discussed in the remainder of the paper.

B. Penetrative Bellman's Equation

For the baseline model, we use Bellman's equation and UCB1 exploration as Algorithm P and Algorithm E, respectively, in Algorithm 1. We compare the results of Bellman's equation and the penetrative Bellman equation as candidates for Algorithm P.

In Table II, the average test cost is presented with the total number of bids as a variable. UCB1 is used as the exploration method, with the exploration parameter α set to 0.1. We can see that when the total number of bids is greater than two, the penetrative Bellman equation outperforms the baseline method by a considerable margin. We note that the baseline model

TABLE I: Comparisons between the average cost of two exploration methods where different values of parameter in each exploration method is tested

layer	2	3	4	5
ϵ-Greedy, ϵ = 0.001	2.9628 ±0.0257	2.7748 ±0.0328	2.7648 ±0.0290	2.7650 ±0.0031
ϵ-Greedy, ϵ = 0.005	2.9582 ±0.0036	2.8201 ±0.0282	2.7773 ±0.0419	2.7510 ±0.0457
ϵ-Greedy, ϵ = 0.01	2.9857 ±0.0227	2.8080 ±0.0113	2.7696 ±0.0179	2.7716 ±0.0305
ϵ-Greedy, ϵ = 0.05	3.0125 ±0.0689	2.8331 ±0.0413	2.8408 ±0.0367	2.8328 ±0.0079
ϵ-Greedy, ϵ = 0.1	3.0575 ±0.0092	2.8758 ±0.0240	2.8679 ±0.0228	3.0035 ±0.1720
no exploration	2.9600 ±0.0372	2.7914 ±0.0080	2.7559 ±0.0616	2.7949 ±0.0358
UCB1, α = 0.05	**2.9329 ±0.0069**	2.7776 ±0.0128	2.7451 ±0.0055	2.7695 ±0.0143
UCB1, α = 0.1	2.9391 ±0.0279	**2.7289 ±0.0595**	**2.6984 ±0.0207**	**2.7397 ±0.0187**
UCB1, α = 0.2	2.9542 ±0.0052	2.8042 ±0.0358	2.7183 ±0.0171	2.7465 ±0.0221

TABLE II: Comparison of the average cost for different methods of updating the Q-functions

Total bids	2	3	4	5
Baseline	**2.9308**	2.8585	2.8795	2.9225
Penetrative Bellman's Equation	2.9329	**2.7289**	**2.6984**	**2.7397**

TABLE III: Comparison of the average cost for our models, those in [2], and Wbridge5

Model	training	validation	testing
layer = 2	2.8013 ± 0.0368	2.9150 ±0.0049	2.9329 ± 0.0069
layer = 3	2.6725 ± 0.0392	2.7363 ±0.0465	2.7289 ± 0.0595
layer = 4	**2.5992 ± 0.0474**	**2.6700 ±0.0245**	**2.6984 ± 0.0207**
layer = 5	2.6442 ± 0.0261	2.7150 ±0.0123	2.7397 ± 0.0187
[2] layer = 4	2.9730 ± 0.0315	3.0697 ±0.0388	3.0886 ± 0.0479
[2] layer = 6	2.9136 ± 0.0384	3.1267 ±0.0092	3.1657 ± 0.0199
Wbridge5	N/A	N/A	3.0039

TABLE IV: Approximate computation time for each model

layer	2	3	4	5
running time per epoch (sec)	121	278	492	713
running time until converge (hrs)	0.5	2.3	6.8	17.9

TABLE V: The average costs of our algorithm coupled with different bidding systems using different ϵ_h

			the coupling bidding system		
Layer	ϵ-human	Wbridge5	SAYC	Natural-5542	CPC
layer = 4	$\epsilon_h = 0$	3.0039	**2.6984**	**2.6984**	**2.6984**
	$\epsilon_h = 0.1$	–	2.7495	2.7074	2.7474
	$\epsilon_h = 1$	–	2.7839	2.7425	2.7776
layer = 5	$\epsilon_h = 0$	–	2.7397	2.7397	2.7397
	$\epsilon_h = 0.1$	–	2.7432	2.7192	2.7425
	$\epsilon_h = 1$	–	2.7850	2.7433	2.7925

has little variance in the performance for varying total bids. This can be attributed to the estimation errors of the Q-value raised by the ordinary Bellman's equation, as stated in Section III. Because the estimation error further accumulates as the total number of bids increases, this cancels out the expected performance improvement that should be gained by increasing the model complexity. We can infer that the primary reason that the penetrative Bellman equation is effective is that it enables the possibility of learning a deep Q-learning model with more bids by providing a more accurate estimate of the cost. The experimental results validate the benefits of the proposed penetrative Bellman equation.

C. Comparison with the State-of-the-Art

We now discuss the experiment with different model structures. We consider the Q-learning model with total bids $\in \{2, 3, 4, 5\}$. For each model structures, we use the validation result to determine the parameter α, where $\alpha \in \{0.05, 0.1, 0.2\}$. We compare the bridge-bidding results with those of that proposed in [2], and the well-known computer bridge software, Wbridge5 [25], which has won the computer bridge championship for the last several years. We run the same 140,000 data instances on our model and that proposed in [2], while running the 20,000 instances of testing data on Wbridge5, using the code provided by the author of [2].

The results in Table III show that the deep reinforcement learning model with layers = 4 achieves the best performance among all models. Moreover, each deep reinforcement learning model outperforms the result achieved by Wbridge5. This showcases that deep reinforcement learning achieves a good result, even with a simple model structure. The result verifies that there is indeed considerable potential for improving the traditional approach to bridge-bidding AI.

D. Computational Time

In this section, we discuss the computing time for training our models. The code is written in MATLAB, and executed on a system with Ubuntu Linux 12.04 LTS AMD64, using Intel Xeon X5560 CPU with 60 GB RAM. We list the training times for one epoch and the total training times for models with different numbers of layers in Table IV. Note that the training time can be further shortened if GPU is utilized.

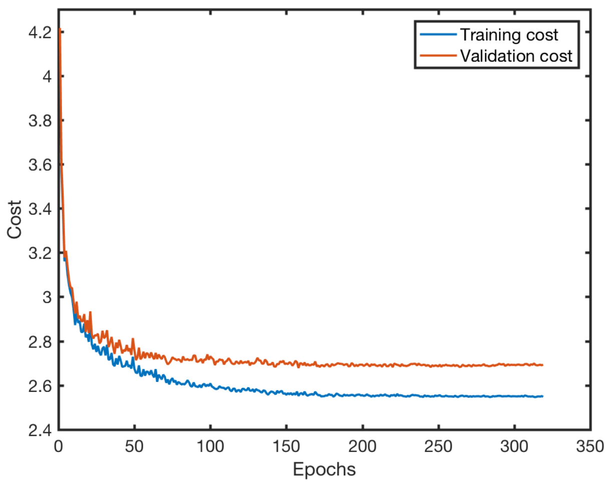
In Table IV, we can observe that the training for models with two and three layers is quite efficient, whereas the training time becomes considerably long for larger models. The total convergence time is approximately to the order of l^3, where l is the total number of layers (or bids) in the model. This is because the complexity of the penetrative Bellman equation has an extra order of l compared with the complexity of Bellman's equation.

E. Comparisons Between Models with and without a Human Opening Bid Incorporated

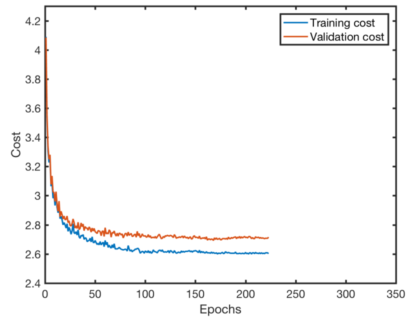
To couple our learning algorithm with a human-designed opening bid, we consider three different bidding systems that are widely used by both amateur and professional players: Standard American Yellow Card (SAYC), Natural-5542, and Chinese Precision Club (CPC). In each of the three bidding

TABLE VI: Opening tables of different learned models.
The abbreviation "bal" refers to a balanced distribution of cards in each suits.

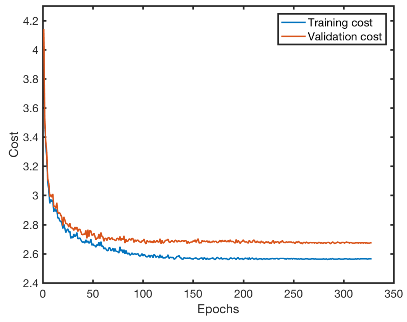
Bid	[2]	ours	ours + CPC	ours + SAYC	ours + Natural-5542
PASS	0-12 HCP	0-10 HCP	0-10 HCP	0-12 HCP	0-12 HCP
1♣	9-19 HCP, 4-6 ♡	11+ HCP	11+ HCP	10+ HCP, 3+♣	10+ HCP, 2+♣
1♢	8-18 HCP, short♠ and 4-6♣	10+ HCP, 5+♡	7-16 HCP, 4+♢	9+ HCP, 4+♢	8+ HCP, 4+♢
1♡	12-23 HCP, w/o long suit	12+ HCP, 5+♠	7-16 HCP, 5+♡	8+ HCP, 5+♡	7+ HCP, 5+♡
1♠	10-19 HCP, 4-6 ♠	16+HCP, bal	7-16 HCP, 5+♠	8+ HCP, 5+♠	7+ HCP, 5+♠
$1NT$	Not used	12+ HCP, 6+♢	14-18 HCP, bal	15-22 HCP, bal	15-21 HCP, bal
2♣	0-17 HCP, long ♣	Not used	10-15 HCP, 6+♣	13+ HCP, long ♡, long ♠	16+ HCP, long ♡
2♢	0-17 HCP, long ♢	Not used	14-22 HCP, long ♠	Not used	7-13 HCP, 6+♢
2♡	0-13 HCP, long ♡	18+ HCP, 5-6♠	Not used	Not used	7-13 HCP, 6+♡
2♠	0-13 HCP, long ♠	Not used	20+ HCP, bal	13+ HCP, 6+♡	Not used
$2NT$	Not used	15-17 HCP, 6+♠	16-20 HCP, long ♡	19+ HCP, short ♠, 3-4 ♡	16+ HCP, long ♠



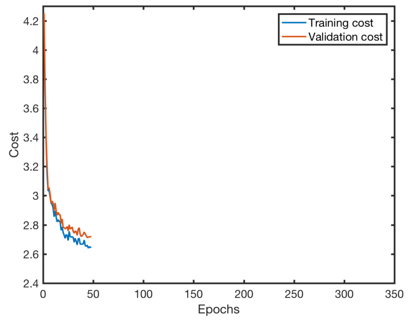
(a) No human opening bid considered, i.e., $\epsilon_h = 0$



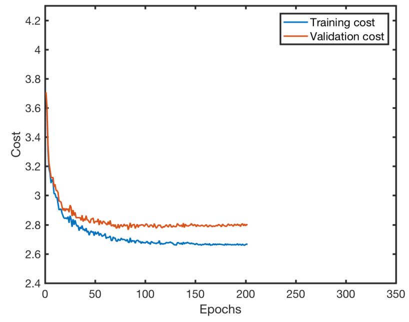
(b) SAYC with $\epsilon_h = 0.1$



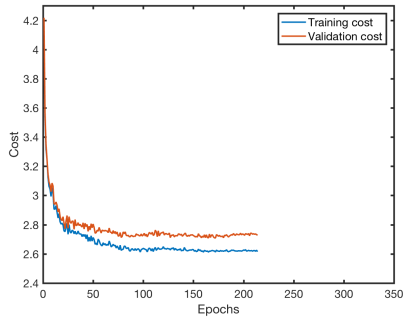
(c) Natural-5542 with $\epsilon_h = 0.1$



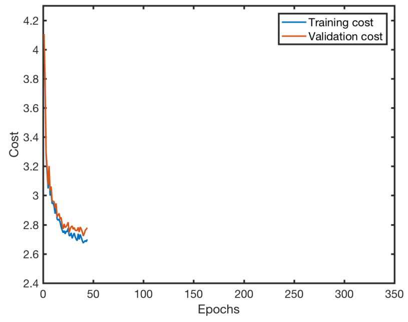
(d) Chinese Precision Club with $\epsilon_h = 0.1$



(e) SAYC with $\epsilon_h = 1$



(f) Natural-5542 with $\epsilon_h = 1$



(g) Chinese Precision Club with $\epsilon_h = 1$

Fig. 2: Learning curves of our model with four layers coupled with different human-designed opening bids using different ϵ_h.

systems, there is a bid called a weak bid, which serves the purpose of interfering with opponents. Because there is no need for a weak bid in the subproblem of bidding with no competitions, we perform experiments on openings without weak bidding for each system. The experiments are conducted on models with four and five layers respectively, with $\alpha = 0.1$ for UCB1. The results are shown in Table V with Wbridge5 as a comparison baseline. To gain further insight into how our learning algorithm reacts to different ϵ_h values as well as different human bidding systems, we further visualize the learning curves of the models coupled with different opening bid rules in Figure 2. In addition, we highlight the differences between the resulting learned models through the qualitative results listed in Table VI, where the opening tables of different models are compared.

From Table V, it can be seen that whether coupling with human-designed opening bids or not, our model outperforms Wbridge5 by a considerable margin. This verifies that our bidding model is not only able to learn effectively, but it also improves existing human bidding systems. Table V also shows that smaller ϵ_h values generally achieves better results under different human bidding systems, indicating that softer combinations do provide more flexibility for our learning model to choose whether or not to leverage the human-designed systems.

In Figure 2, it is shown that different human-designed opening bids that our model is coupled with have very different effects on the learning curve. There is a particularly notable contrast between natural bidding systems and precision bidding systems. For natural bidding systems, such as SAYC and Natural 5-5-4-2, the models usually take longer to converge. On the other hand, we can see that the models combined with the precision club opening converge a lot faster than the others, although they achieve slightly inferior performances. Interestingly, the results are in line with the philosophies that are used to develop these human bidding systems. For example, precision club systems are designed to be more efficient and precise, so as to let the partner immediately knows the potential hand of the opener. This advantage of the precision club enables our model to learn faster, which is reflected in the short learning curves. While being more precise, precision club enforces a stronger limitation on the choices of opening bids. This may explain the limited performances of the models combined with the precision club opening.

Moreover, from the qualitative results shown in Table VI, it can be seen that when coupled with different human bidding systems our algorithm indeed converges to different models and exhibits different opening strategies. In addition, the resulting models somewhat resemble their corresponding human bidding systems. For example, the model coupled with CPC has tighter constraints on the high-card points compared to the others, matching the characteristics of the precision club opening, and the models coupled with SAYC and Natural-5542 have similar rules (the suit length) for the one-level opening bids as their original counterparts.

As the learning curves and the resulting opening tables behave in line with the characteristics of different human bidding systems, this finding reveals that human knowledge can indeed be incorporated into our learning algorithm, regardless of the benefits in terms of the final performance.

TABLE VII: The average costs with different levels of hand features used as the extra dimensions in the training objective

	feature			
Layer	w/o feature	lowest	median	highest
layer = 4	2.6984	2.7225	2.6910	**2.6850**
layer = 5	2.7397	2.7450	2.7175	**2.6950**

F. Comparisons between Models Trained with and without Human-Crafted Hand Features

In these experiments, we consider three levels, namely the lowest, median, and highest, of hand features to be added to the output layers of the model. For the lowest-level features, there are 20 dimensions, where four of these represent the length of each suit, and the other 16 are binary values representing the presence of the honors in each suit. For the median-level features, we leverage the point-count system (4-3-2-1 count for the honors) used in a human bidding system to compress the 16-dimensional raw high-card features into a single dimension *HCP* representing the strength of the whole hand. Finally, for the highest-level features we extend the median-level features with four extra dimensions representing the quality score for each suit. The experiments are conducted with models consisting of four and five layers respectively, where $\alpha = 0.1$. The corresponding results are shown in Table VII.

From Table VII, we see that a better performance can be obtained by adding more sophisticated hand features to the output layer while training the bidding system. Specifically, from the lowest-level features to the median-level features, we introduce the use of the human-designed point-count system, aiding the algorithm with the concept of high-card strength. From the median-level features to the highest-level features, we further provide the algorithm with quality scores, which measure the strength of each suit. The results hint that these human-crafted hand features can indeed improve the learning algorithm, and demonstrate yet another way to incorporate human knowledge into the learned bidding system.

In order to give a more concrete idea of the result, we randomly choose five examples in the testing data, where the bidding continues for at least four turns, and demonstrate their resulting predictions for median-level hand features. After Player 2 received the third bid (which was bid by Player 1), we compared the estimation of the feature of Player 1 by Player 2 with the actual feature of Player 1. The result is shown in Table VIII, and demonstrates that Player 2 is able to precisely estimate the hand of Player 1, even when Player 1 has only made two bids.

V. Conclusion and Future Works

We have proposed a novel model that automatically learns to bid from raw hand data by coupling deep reinforcement learning with improved exploration and update techniques. To the best of our knowledge, our proposed model is the first to

TABLE VIII: Five examples, where features of Player 1 are listed in the actual column, and the estimations of Player 2 are listed in the estimate column. The bidding history, along with the best bid and cost, is also listed in the table.

	actual	estimate	actual	estimate	actual	estimate	actual	estimate	actual	estimate
number of spades	5	5.1024	2	2.2202	3	3.1463	4	4.5931	3	2.7333
number of hearts	2	2.2983	4	4.9368	4	3.5169	3	2.5334	1	1.0620
number of diamonds	3	2.3366	5	2.9966	4	3.8936	2	3.0599	6	5.9515
number of clubs	3	3.2061	2	2.9111	2	3.3676	4	2.8925	3	3.2824
HCP	5	4.9745	21	19.5970	9	9.7680	19	18.5185	5	5.7930
bidding history	P-1NT-2♠-4♡		1♠-1NT-3♠-4♡		P-1♣-1NT-1NT		1♠-2♣-5♡-6♡		P-1♡-2♢-2♢	
best contract	4♡		4♡ or 3NT		2♢		7NT or 7♡		2♢ or 3♢	
cost(IMP)	0		0		4		11		0	

tackle automatic bridge bidding using raw data without additional human knowledge. We demonstrated that our proposed model outperforms champion-winning programs and state-of-the-art models by a considerable margin. This superior performance validates the potential of deep learning for achieving a competitive bidding system on its own.

In addition to learning entirely from data, we further demonstrated how human knowledge can be leveraged by the proposed framework. Experimental results show that our model can successfully take advantage of existing human knowledge to obtain a better performance, shedding light on the potential of the combination of human knowledge and a self-learning AI. We also believe that the qualitative results can facilitate interesting comparisons between the self-learning AI and human players.

One important future direction is to extend our model for the other subproblem of bidding with competition. In particular, the flexibility of the proposed model allows it to improve its bidding strategy, with or without competition, by self-playing as its own opposing team, or by playing with other human or AI teams. One immediate difficulty for self-playing is to identify a good exploration strategy for bidding with competition.

Acknowledgment

We thank the anonymous reviewers for valuable suggestions. We also thank Chun-Yen Ho and other members of the NTU CLLab for insightful discussions. This material is based upon work supported by the Air Force Office of Scientific Research, Asian Office of Aerospace Research and Development (AOARD) under award number FA2386-15-1-4012, and by the Ministry of Science and Technology of Taiwan under number MOST 103-2221-E-002-148-MY3 and MOST 106-2119-M-007-027.

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[2] C.-Y. Ho and H.-T. Lin, "Contract bridge bidding by learning," in *Proceedings of the Workshop on Computer Poker and Imperfect Information at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[3] T. Sandholm, "The state of solving large incomplete-information games, and application to poker," *AI Magazine*, vol. 31, no. 4, pp. 13–32, 2010.

[4] N. Yakovenko, L. Cao, C. Raffel, and J. Fan, "Poker-cnn: A pattern learning strategy for making draws and bets in poker games using convolutional networks," in *the AAAI Conference on Artificial Intelligence*, 2016.

[5] M. Ginsberg, "Gib: Steps toward an expert-level bridge-playing program," in *International Joint Conference on Artificial Intelligence*, 1999, pp. 584–593.

[6] G. L. Carley, "A program to play contract bridge," Ph.D. dissertation, Massachusetts Institute of Technology, Department of Electrical Engineering, 2004.

[7] T. Lindelof, *COBRA: the computer-designed bidding system.* v. Gollancz, 1983.

[8] A. M. Stanier, "Bribip: A bridge bidding program," in *International Joint Conference on Artificial intelligence*, 1975, pp. 374–378.

[9] A. Wasserman, "Realization of a skillful bridge bidding program," in *Proceedings of the Fall Joint Computer Conference*, 1970, pp. 433–444.

[10] B. Gambäck, M. Rayner, M. Yard, and B. Pell, "Pragmatic reasoning in bridge," 1993.

[11] A. Amit and S. Markovitch, "Learning to bid in bridge," *Machine Learning*, vol. 63, no. 3, pp. 287–327, 2006.

[12] L. L. DeLooze and J. Downey, "Bridge bidding with imperfect information," in *IEEE Symposium on Computational Intelligence and Games*, 2007.

[13] C.-K. Yeh and H.-T. Lin, "Automatic bridge bidding using deep reinforcement learning," in *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI)*, September 2016, pp. 1362–1369.

[14] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[15] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[17] M. Riedmiller, "Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method," in *European Conference on Machine Learning*, 2005, pp. 317–328.

[18] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," *arXiv preprint arXiv:1602.04621*, 2016.

[19] I. Osband and B. Van Roy, "Bootstrapped thompson sampling and deep exploration," *arXiv preprint arXiv:1507.00300*, 2015.

[20] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[21] W. Chu, L. Li, L. Reyzin, and R. E. Schapire, "Contextual bandits with linear payoff functions," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 208–214.

[22] J. Langford and T. Zhang, "The epoch-greedy algorithm for multi-armed bandits with side information," in *Advances in neural information processing systems*, 2008, pp. 817–824.

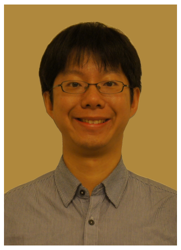
[23] W. Li, X. Wang, R. Zhang, Y. Cui, J. Mao, and R. Jin, "Exploitation and exploration in a performance based contextual advertising system," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010, pp. 27–36.

[24] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," *CoRR*, vol. abs/1611.05397, 2016.

[25] Y. Costel, "Wbridge5 bridge software," 2014.

Chih-Kuan Yeh received a B.S. in Electrical Engineering from National Taiwan University in 2016, where he worked on automatic bridge bidding with Prof. Hsuan-Tien Lin. Currently, he is pursuing his Ph.D. in Machine Learning from Carnegie Mellon University under the supervision of Prof. Pradeep Ravikumar. His research interests include understanding how deep learning algorithms work and their real-world applications from both theoretic and emperical aspects, and developing efficient and interpretable algorithms towards scalable and explainable deep learning.

Cheng-Yu Hsieh received a B.B.A. in Information Management from National Taiwan University in 2016, where he worked on investigating optimization for large-scale scheduling problem with Prof. Ling-Chieh Kung. Currently, he is pursuing his M.S. in Computer Science and Information Engineering from National Taiwan University under the supervision of Prof. Hsuan-Tien Lin. His research interests include studying on fundamental machine learning tasks, designing learning algorithms towards practical needs, and exploring the potential of human-machine collaboration.



Hsuan-Tien Lin received a B.S. in Computer Science and Information Engineering from National Taiwan University in 2001, an M.S. and a Ph.D. in Computer Science from California Institute of Technology in 2005 and 2008, respectively. He joined the Department of Computer Science and Information Engineering at National Taiwan University as an assistant professor in 2008, and was promoted to an associate professor in 2012, and has been a professor since August 2017. He is currently also the Chief Data Scientist of Appier, a startup company that specializes in making AI easier in various domains, such as digital marketing and business intelligence.

From the university, Prof. Lin received the Distinguished Teaching Award in 2011 and the Outstanding Mentoring Award in 2013. He co-authored the introductory machine learning textbook "Learning from Data" and offered two popular Mandarin-teaching MOOCs "Machine Learning Foundations" and "Machine Learning Techniques". His research interests include mathematical foundations of machine learning, studies on new learning problems, and improvements on learning algorithms. He received the 2012 K.-T. Li Young Researcher Award from the ACM Taipei Chapter, and co-led the teams that won six KDDCup world champions between 2010 and 2013.