Analysis of selected reinforcement learning applications in contract bridge

R. JAROSZ

robert.jarosz@wat.edu.pl

Military University of Technology, Faculty of Cybernetics
Kaliskiego Str. 2, 00-908 Warsaw, Poland

This paper presents an overview of four selected solutions addressing problem of bidding in card game of contract bridge. In the beginning the basic rules are presented along with basic problem size estimation. Brief description of collected work is presented in chronological order, tracking evolution of approaches to the problem. While presenting solution a short description of mathematical base is attached. In the end a comparison of solution is made, followed by an attempt to estimate future development of techniques.

Keywords: artificial intelligence, bidding, bid prediction, contract bridge, game theory, incomplete knowledge, machine learning, neural networks, Q-learning, reinforcement learning, supervised learning.

DOI: 10.5604/01.3001.0053.9702

1. Introduction

Reinforcement learning is an area of machine learning, particularly useful in solving game theory problems. The learning process involve numerous game solving attempts against interactive *environment*. Generally, learning process does not need prepared data samples as games can be generated on runtime and reward is calculated and delivered by *environment.* This method proved very efficient (on super human level) in games complicated enough to render classic game theory methods (e.g. backward induction) [1], [2] computationally impossible. An example of such application is an algorithm AlphaGo Zero developed by DeepMind team[3].

In this paper we focus on a popular card game of contract bridge. The article presents an overview of some chosen machine learning approaches.

2. Contract Bridge

The game involves four players in two teams, players are commonly named after directions and placed in front of his/her partner (teams: East-West and North-South). Initially players do not know each other's cards (except for 13 cards in their own hand). The game consists of two phases. First one is bidding phase and a second follows playing phase. In the first phase players bid the contract which the winning side must fulfil later in the playing phase.

3. Playing Phase

Although being second in order, the contract phase is significantly important in planning actions in the first place. Contract declarer's partner is called dummy and after first of defenders places his card for the first trick, dummy player places all his cards visible on the table. All his cards are now managed by declarer. Playing phase consists of 13 tricks for which every player adds card to the trick (declarer places one card from hand and one from the table). If possible, every placed card must be in the same suit as first card placed in the trick. In contract with trump, a highest card in trump suit wins the trick, in no trump contracts, the highest card in suit of first card placed in trick wins it. Each trick is resolved and a winner is selected to gain the trick. After thirteen tricks resolved game both side have collected a number of tricks (which sum to 13). In this part of the game optimalisation target is easy to define – it is maximalization of collected tricks. This problem is often regarded as solved – with optimal strategies already known. Commonly players utilize Double Dummy Analysis [4], which provides upper bound of maximal cumulative reward returned after continuing policy under assumption of complete knowledge about cards. As pointed in [5], [6] although the assumption of complete knowledge is strong, it is reported that accuracy of analysis resembles actions of professional players well and more efficient than an actual play.

4. Bidding Phase

On a competitive level, this phase is considered a core of the game – players' skills are distinguished here, as on professional level players perform very similarly in the playing phase. Problem of solving this phase is still open for improvements.

Players of one team cooperate to bid a contract that is possible to fulfill with cards in their hands. The higher is the contract, the higher gets reward for fulfilling it, however penalties for possible failure depend on how many declared tricks were not taken (called undertricks). The contract consists of number from 1 to 7 and trump from ♣, ♦, ♥, ♠, NT (where NT means "no trump" which is selected by players considering their cards strong in every trump). The number n in contract obliges declarer to take $n + 6$ tricks in the following playing phase in order to fulfill the contract. Bid trump marks privileged suit (or that there is no such suit) in following playing phase. As for bidding phase, it is important to point out that bids are ordered from lowest to highest in following manner: $[1\clubsuit, 1\diamondsuit, 1\heartsuit, 1\spadesuit, 1\text{NT}, 2\clubsuit, \ldots, 2\text{NT}, \ldots, 7\text{NT}]$.
While bidding, players have following possible actions to perform:

- Bid;
- Double or Redouble;
- Pass.

Players can bid a contract higher than one currently being bid. The first player in pair who has bid a trump will be declarer in case his/her axis wins bidding with this trump. In case of bidding higher contract, one can double the bid of directly previous bidder. Doubling does not change contract, however doubles rewards and penalties if the current contract manages to be final. Similarly directly after doubling, the enemy side can redouble the contract, doubling scores once more. Player can pass if he/she does not desire to make any bid or doubling/redoubling. Beside doubling, rewards are also affected by the state of vulnerability. This state is independent for axis North-South and East-West, in sport bridge this state is pre-set for the deal. If axis is in vulnerable state, it's bonus rewards and penalties are increased, typically (but not always) doubled.

After setting a contract and playing it, the contract is checked if it is fulfilled and points for it are rewarded according to the table of rewards. Overbidding (not fulfilling the contract) is especially painful for declaring team as not only they do not get points for contract, but their opponents are granted bonus points for every undertrick.

Players' calls in bidding are the only legal information carriers about cards in their hand. In a natural bidding system, players bid in suits they want to play contract. Professional players use more advanced bidding systems, aiming at giving more precise card information. This bids should be understood by partner and the bidding system must be explainable to opposing players. Systems used by human players usually utilize evaluating cards with points for high figures and high numbers of cards in particular suit. Model of learning algorithm using human cards' evaluation was used in [7].
A challenging problems of bidding system are:

- Decreasing number of possible bids (subsequent bids must be higher than previous);
- Limited action number before rising risk of overbidding;
- Enemy team making their own bids further reducing possible bids.

The information about one's cards is crucial for partner in bidding phase. As the playing phase starts, every player knows the dummy's cards, making declarer aware of all his/her teams cards. Each defender however know his/her cards and dummy's cards which makes half of their team's cards and half of enemy team's cards. The information if certain unseen card is in hand of defender's partner's or declarer's hand remains uncertain (however it may be hinted from history of bidding).

Generally bid calls are made in order to [8]:

1. Ask questions (e.g. if your hand has a trait A answer with bid X, answer Y if it has trait B);
2. Answer – provide answer to previously asked question;
3. Assert – state about one's hand's property;
4. Deny – state about lack of certain property in one's hand;
5. Interrupting – interfering in opponents communicating.

5. Hands assessments

It is common in human techniques to evaluate their hands to point values indicating general strength of the cards in hand. Example hand evaluation follows:

- 7 points for each Ace,
- 4 points for a King,
- 3 points for Queen,

- 2 points for Jack if one has two higher figures in the same suit,
- 4 points for having a single card in one suit (if it is also a high figure, points do not cumulate),
- 7 points if one does not have any card in certain suit.

Presented points values are not obligatory standard and every team can have their own evaluation as far as their bidding system is clear for the opponents. Such card evaluation is not necessary, as team can base their bidding system on raw card sets. Omitting this preprocessing is hard for human players, however computer agents using neural networks may use raw card data as an input.

6. Basic Bridge Combinatorics

Below there are presented some basic combinatoric facts about bridge game.

Total number of possible deals:

$$|\Omega| = \binom{52}{13}\binom{39}{13}\binom{26}{13} \approx 5.36 \cdot 10^{28} \quad (1)$$

Number of possible deals from perspective of player having his cards already distributed:

$$|R_H| = \binom{39}{13}\binom{26}{13} \approx 8{,}44 \cdot 10^{16} \quad (2)$$

Number of possible hands of partner, ignoring the distribution remaining cards in hands of opponents:

$$|P_H| = \binom{39}{13} \approx 8{,}12 \cdot 10^{9} \quad (3)$$

7. Bridge bidding as contextual bandit problem

One of the approaches shown by Chun-Yen Ho and Hsuan-Tien Lin in [7] tries to teach AI bidding in bridge assuming passive play of opponents. The algorithm treats bidding as contextual bandit problem [9] and utilizes UCB algorithms [10] to solve it.

Authors noted that for every random deal it is possible to make double dummy analysis and how many tricks can be taken for every possible contract. Having a dataset of length N of learning deals, it is possible to extract cards of partners (without loss of generality named North and South): x_N, x_S. Secondly, for this deals a cost vector $c = [c_{1\spadesuit}, \ldots, c_{7\mathrm{NT}}]$ can be derived using double dummy analysis for very possible contract. This analysis show not only which contract is best suited for this deal, but also how other contracts are worse.

Firstly authors introduce model for assessing their solution by creating cost-sensitive classification problem (CSC) [11] for bid sequence. For purpose of assessing baseline and upper bounding, cheating methods are created and discussed.

Baseline method assumes that only a single bid is made, and the partner (South) always says PASS, giving no information about his cards. Baseline strategy uses learning classifier mapping player's hand x_N to bid b_N. Classifier learns by minimalizing statistical cost of bid $c \circ b_{base}(x_N)$. Such classificatory maps one player's hands to statistically optimal bid with assumption that no information about any other hands is known.

The second reference method tries to establish upper bound of performance to this model. Once again, they consider a single-bid sequence, however assuming that bid maker knows not only his/her cards, but also partners. This models a hypothetic situation, trying to answer a following question: "If bidding system is capable of sharing complete information of players' cards, what is an optimal bid?". Such an assumption is surely not possible for real games and having such knowledge would be considered cheating. In this case however presented method is used as a tool to measure performance of proposed later strategy. Classifier learns to map two cooperating players' hands to statistically optimal bid minimalizing function $c \circ b_{upper}(x_N x_S)$.

Both reference mode use CSTSR (cost-sensitive two-sided regression) and CSORS (cost-sensitive one-sided regression) [12] to solve defined CSC problems.

Authors of the research defined shared strategy as G:

$$G: X \times X \to \mathcal{B} \quad (4)$$

predicting next bid based on two players' cards, where $\mathcal{B}$ is a set of legal bids. For one player the function g predicting bids is defined as:

$$g: X \times \mathcal{B}^k \to \mathcal{B} \quad (5)$$

Function g considers player's cards and sequence (history) of bids so far. In notation used by authors b^l means vector of l bids since the beginning and $b[l]$ means l-th bid. Function g must follow particular rules:

$b[1] = g(x_1, \emptyset), b[2] = g(x_2, b^1), \ldots \quad b[l] = = g(x_l, b^{l-1})$. Moreover, bids must be ascending: $b[1] < b[2] < \cdots < b[l]$ and $b[k] \neq PASS$ for $k > 1$ and $k \leq l$.

For learning algorithms, they have chosen UCB and consider possible bids as arms of bandit machine in contextual bandit problem. They proposed to represent bidding history as a node V in a graph. The graph may be tree shaped or organized in layers, and can be explored only in one direction – from root to leaves or top layer to bottom (in layered graph). Nodes of the graph are labeled with bids from $\mathcal{B}$, and children of nodes are possible bids made from parent node. In layered model node 1♦ could be linked upwards with PASS node and 1♣ node in the same time, while in tree model 1♦ has to separate instances.

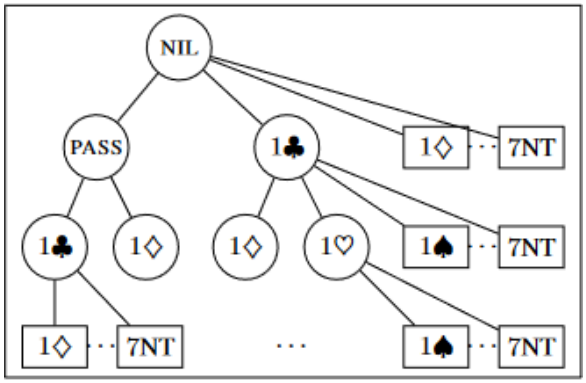

Fig. 1. Tree graph of V-nodes in bidding history

Each of such nodes is an independent classifier $b(x)$. The advantage of such approach is reducing the input of classification function – classifier r is bound with node and a historical context (bidding sequence) of a state is built in the classifier. Another advantage is brough by limiting possible actions (e.g. from 1♦ there is no child node of 1♦ or 1♣). The disadvantage of presented solution is an expanding number of needed classifier as for every node one is needed. This problem made authors limit bidding length to 5 bids.

After selecting bidding sequence, it's final bid (contract making) was used to evaluate that sequence. Then starting from leaf node policy was updated for visited nodes (in order from leaf to root). However, authors diagnosed problem with UCB – top layers could be explored many times in comparison to lower layers. As a solution they suggested a penetrating update. Whenever model would output an early pass, another bid would be selected to continue bidding. Node directly leading to early pass is updated conditionally regarding some probability p – in contrast to normally processed nodes, which are always updated.

In experiments conducted in the research, two particular UCB algorithms have been tested, namely LinUCB[13] and UCB1[10] with different parameters. The results were compared on validation and test datasets (disjoint with training dataset) with winner of several annual computer bridge competitor WBridge5 [14]. Using presented experiments, authors showed a similar performance to WBridge5, especially in tree graph model with 6 layers (5 bids) and the UCB1 algorithm. Although the solution is for now limited in (effective) number of layers and experiment assumed passivity of opponents, the results are indeed promising. It is worth to note that authors pointed and addressed difficulty of modeling varying length of bidding history. As a direction of future research, authors pointed problem of competitive bidding (with active opponent) and extending layered model for better handling potential slam and big slam deals.

8. Approach with Q-learning

In this section, the approach of Chih-Kuan Yeh and Hsuan-Tien Lin [5] is presented. Authors noted that the most of AI players mimic human systems, which naturally makes them less efficient. Intuitively as in real world, learning from others without self-study and research is not a way to surpass a teacher. In addition, authors points out another problem. Human bidding systems contain rules of how to bid based on cards owned by one player and bids made by partner. Following the rules often leads to ambiguous situations and the final choice is left for the players to made. A single bid (unless it is a last significant bid) is difficult to assess properly in the exact moment. Its value is not determined by contract it proposes, it is just a part of information exchanged between partners. A good contract may be made after a successful information exchange and still previous bids standalone could propose poor contracts. This is challenging problem for reinforcement learning – to assess bids as communication medium rather than as contract proposal. The problem becomes more difficult as opponent team makes their own bids and interfere in communication by locking some call.

Similarly to the previous attempt, researchers limited problem to cooperative bidding without interference of opponents. In such model opponents always call PASS and cooperating agents may call PASS as a first bid.

Authors used popular in reinforcement learning approach of Q-learning [15], [16]. Method of Q-learning is based on concept of Bellman equation and it's estimation called Q-function:

$$Q: \mathcal{S} \times \mathcal{A} \to \mathbb{R} \tag{6}$$

Q-function returns reward $r = Q(s,a)$ for performing action a in state s. In game model, proper action is selected according to recursively defined optimal Q-function:

$$Q^*(s,a) = E\left[r + \gamma \max_{a'} Q^*(s',a')\right] \tag{7}$$

Function determines maximal reward for action a made in state s, it sums instant reward r and result of Q^* for next state s' with respect to discount factor γ determining if future rewards are relatively more or less important (deflation or inflation). Determining Q^* function is generally a difficult task. In situation of game with not complete knowledge it is simply impossible, therefore it is convenient to use approximation of this function. One can imagine Q-function as an array of $|\mathcal{S} \times \mathcal{A}|$ entries with some values (usually initiated randomly), while learning algorithm explores states and actions. Every time action a in state s is exercised, algorithm updates entry corresponding to s and a – based on current belief of next state s' and optimal (believed to be) action a' available at that state.

$$\begin{aligned} Q_{new}(s,a) &= (1-\alpha)Q_{old}(s,a) + \\ &+\alpha\left(r_t + \gamma \cdot \max_a Q_{old}(s',a)\right) \end{aligned} \tag{8}$$

In the above equation factor α determines intensity of learning. Q-function is often approximated with neural network, as maintaining such an array of rewards is often incredibly memory consuming. In this case it is similar. Neural variant of Q-function is denoted $Q(s,a,\theta)$, where θ is weights in neurons of the network.

Input to the neural network uses representation of state which is known to agent as list of cards (array of Boolean values with length 52) and array of bools corresponding to possible bids from ordered set $\{PASS,\ 1\clubsuit, 1\blacklozenge, 1\heartsuit, 1\spadesuit, 1\text{NT}, 2\clubsuit, \ldots, 7\text{NT}\}$. With an assumption of passive opponents and constraint that subsequent bids must be higher than previous $a^{(t)} \geq a^{(t-1)}$, it is possible to represent every possible history of bids in such array. Every odd 'true' corresponds to bids of the first player and every even 'true' to bids of the second player. Although this history representation is compact and convenient in this model, it does not suit bidding model with bidding opponents – which is not necessary for subproblem addressed by researchers.

The concept of Q-learning was further modified to omit following steps and include only the estimation of the final bid. With instant reward $r = 0$, simplified Bellman equation for i-th sequential bid would be:

$$Q^{(i)}(s,a) = \max_{a'} Q^{(i+1)}\left(s^{(1')}, a^{(1')}\right) \tag{9}$$

Meanwhile proposed penetrating equation addresses last bid t-th since i.

$$Q^{(i)}(s,a) = \max_{a'} Q^{(i+t)}\left(s^{(t')}, a^{(t')}\right) \tag{10}$$

Q-functions based on both equations where tested in performed experiments.

Reinforcement learning has also an exploitation-exploration tradeoff problem, which is approached with ϵ-greedy algorithm and UCB1. In ϵ-greedy strategy with probability of ϵ algorithm chooses random action (explores) and with $1-\epsilon$ probability agent takes action with the most promising final reward. In comparison, algorithm UCB1 dynamically adjusts to exploit-explore dilemma.

$$a_p = \arg\max_a \left[Q(s,a,\theta) + \delta\sqrt{\frac{2\ln T}{T_a}}\right] \tag{11}$$

Where T is the number of total examples used to learn agent so far and T_a is number of times action a was taken before. This first element of sum tends to select action believed to be good (exploit) and the second element favors rarely explored actions. Parameter δ (often written Greek α causing confusion with a) is used to determine balance between exploiting and exploring – lower δ makes exploiting more probable.

In their experiments researchers have shown that Q-function based on penetrative Bellman equation performed better than a single step counterpart, especially for the bid sequences longer than 2. Authors suggested that penetrative variant reached better results due to more accurate estimation of costs (rewards for particular contracts). In addressing exploit-explore tradeoff problem, UCB1 was better than ϵ-greedy strategy.

Finally authors compared trained agent with work of Chun-Yen Ho and Hsuan-Tien Lin

in [7] and champion WBridge5 [14], showing that their model performs better in this problem. Researchers have built not only successfully learning agent for bidding using only self-play in bidding phase, but also have shown that such agent can have competitive accuracy in bidding. Interestingly, they also managed to provide bidding opening table for their algorithm.

The work is yet another step in improving artificial intelligence possibilities in problems of game theory. For the next steps in this subject, authors suggest extending to subproblem of bidding with competition.

9. Competitive Bridge Bidding

Next interesting approach was presented by Jiang Rong, Tao Qin and Bo An in [17]. One of the differences between this approach and previous ones is not assuming passivity of the opponents, moreover, this model assumes using calls like double and redouble.

For the purpose of taking a note of opponents' bids, a different representation of history is proposed. For every contract bid (for example 1♥) there are at most 8 following non-contract calls positioned in order *pass-pass--double-pass-pass-redouble-pass*. Therefore, for every contract bid, history array has 9 Boolean values to represent it, every bit is set to 1 if a call was made or to 0 otherwise. First value is set to determine if contracting bid has even been called and following bools are set if according not contract call was made. Example bidding of 1♥-pass-pass-double-redouble would be written in sequence: $1_{1\heartsuit}1_{\mathrm{p}}1_{\mathrm{p}}1_{\mathrm{d}}1_{\mathrm{rd}}0_p0_p$. If certain contract bid has not been made, it is represented in history with 9 zeros. Before first contract bid is declared, there can be at most 3 subsequent passes, so final bid history would be represented as bits array:

[3 opening values] + [9 values for 1♣] +
+ [9 values for 1♦] + … + [9 values for 7NT]

This make history space $H = \{0,1\}^{318}$, with a note that not every value is valid, since player cannot be doubled by his/her partner (e.g. slice of 1 10 1 00 0 00). Note that there is no need of marking which call was made by which side as calls are made in strict orders player who placed 1 in bit of history can be determined by the number of previous 1s in history.

To represent cards owned by player, the vector c of 52 values with each value corresponding to one card in deck – 13 of them are set to 1, the rest of them are 0. Hand space can be defined as:

$$C = \{c_i\}^{52} \,|c_i \in \{0,1\} \wedge \sum_{i=1}^{52} c_i = 13 \tag{12}$$

Vulnerability is represented by two bit values: "00" meaning no vulnerability, "01" meaning vulnerability of opponents, "10" of player's team, and "11" means that both teams are vulnerable.

Solution is based on two neural networks: *estimation neural network* (ENN) and *policy neural network* (PNN). Player p wants to estimate his partner's hand with function:

$$D\colon C \times V \times H \to [0,1]^{52} \tag{13}$$

Function given by network returns estimated probability of player's partner p^+ holding cards from set of C_{A-c_p}. $D(c_p, v, h)[i]$ represents probability of p^+ holding i-indexed card. Although c_p and v are known and easily understandable for player p, the meaning of h is problematic. The core problem of calculating D-function is calculating conditional probability of player p^+ having cards c_{p^+} provided that he made bids $B_i(p^+, h)$ – bids made by p^+ in history h. Assuming players have a policy function:

$$\sigma\colon C \times V \times H \times [0,1]^{52} \to [0,1]^{38} \tag{14}$$

Such policy given player's cards, vulnerability vector, history vector and assumption abut partners ($D(c_p, v, h)$) card returns vector of probabilities for calling each of 38 possible calls. Given policy function and it's input, it is straightforward to deliver resulting call. Given the resulting call of player p^+, his policy σ and inputs v, h, it is theoretically possible to infer distribution of c_{p^+} using Bayes theorem for conditional probability. However, due to large number of possible combinations of c_{p^+}, namely $\binom{39}{13}$ it is computationally not feasible. However, in this place neural network estimation can be used replacing D-function with ϕ_ω, where ω denote neural network parameters. ϕ_ω works as ENN, and second neural network σ_θ is PNN.

The process of learning algorithm started with ENN as its output is one of the inputs in PNN. Firstly ENN was trained with data collected from tournaments played by professional bridge players. An archive was

created thanks to Vugraph Project [18]. Trained ENN is used to train PNN as preprocessing of input data. After both networks are trained using supervised learning with expert data, both neural networks are trained further with self-play in the model of reinforcement learning. Games are played between two teams of learning algorithm, both side trying to improve their policy. Every deal is played twice by algorithm – once in position of North-South and second one in position of East-West. During training, updated networks are added to policy pool from which opponents are picked. This method ensures that algorithm learns to play against different (evolving) opponents.

Experiments made by authors show that reinforcement learning indeed improved performance of algorithm and adding assumptions generated by ENN improves PNN. Researchers pointed out that provided expert data was rich source of information and calls on lower level, however it provides significantly less data for higher number bids. Because of this feature of data, reinforcement learning had space for improvement as it could exercise situation rarely spotted on tournaments. Beside comparing their solutions with each other, they forced their best algorithm (reinforcement learning improved PNN+ENN) to play against Wbridge5 [14]. RL-PNN+ENN outperformed Wbridge5 significantly – scoring 0.25 IMP (International Match Points).

Authors stated plans for further improving algorithm, releasing it for public tests and interestingly to propose translation from network based system to convention for humans.

10. Deal – Minimal scope agents

Another interesting approach was presented by Xiaoyu Zhang, Wei Liu, Linhui Lou and Fangchun Yang [19]. In this solution authors made effort to create algorithm able to explain its policy to human players, which is mandatory in human tournaments. In sport bridge players must unveil their bidding strategy and opponents have rights to ask for explanation of steps taken in case they have doubt.

In solution, the *recurrent neural network* (RNN) is used. Such network differs from traditional feedforward network in neuron connection. In feedforward network the signal is propagated through layers of neurons from first to last layer and represent function of $y = \theta(x)$. Such network has no internal memory and inserting to subsequent inputs resolves in two unrelated outputs, while RNN is capable of using sequences of input data. A layer of network accepts external input and set of activation vector and produces output alongside activation vector for further use. Activation vector plays a role of internal state of the network. With incoming next slice of input data, it is combined with previously achieved activation vector and fed to the same layer of network. It is often presented in unveiled form, resembling feedforward network, however following layers share the same weights. Recurrent layer models function $(y_i, h_i) = = \omega(x_i, h_{i-1})$, with x_i being i-th input starting with $i = 1$, h_i being hidden output – activation (state) after feeding with i-th input (h_0 is not data dependent) and y_i being observable output. Subsequent executions form equations:

$$\begin{aligned} (y_1, h_1 &= \omega(x_1, h_0) \\ (y_2, h_2 &= \omega(x_2, h_1) \\ &\dots \\ (y_t, h_t &= \omega(x_t, h_{t-1}) \end{aligned} \tag{15}$$

In bidding model authors composed input vectors x_i of card hand representation X_i, last bidding action b_L, vulnerability v and position p ($p = 1$ if it is player's time to bid, 0 otherwise). Concatenated input is fed to stacked RNN realizing function:

$$s = F_s(X_i, b_L, v, p) \tag{16}$$

Note that notation $F_s(X_i, b_L, v, p)$ is logically equivalent of defined $\omega(x_i, h_{i-1})$, utilizing hidden vector h (which is part of output s and state parameter of stateful function F_s).

Simultaneously, output of the RNN network is passed to densely connected neural network MLP (Multi-Layer Perception) to classify hand (extract features) and predict next bid.
MLP network outputs 4 handful vectors:

$$\begin{aligned} s_H &= F^H_{mlp}(s) \\ s_{C^{suit}} &= F^{C^{suit}}_{mlp}(s) \\ s_Y &= F^Y_{mlp}(s) \\ s_G &= F^G_{mlp}\left(s, s_H, s_{C^{suit}}, s_Y\right) \times P^B_{mask}(L) \end{aligned} \tag{17}$$

where s_H is probability distribution of partner's HCP (high card points) – sum of point evaluation for powerful cards: A, K, Q, J. In evaluating system used in model HCP is a natural number in $0..=40$, therefore s_H is 41 float number vector. Vector $s_{C^{suit}}$ is probability distribution of number of cards ($0..=13$) hold by partner in each suit (4×14 float numbers) and s_Y is probability distribution

of cards in other 3 players' hands (3 × 13 float numbers). With these 3 game state prediction vectors and internal state of RNN a bid prediction s_G is calculated with regards to mask $P^b_{mask}(L)$ zeroing probability of making illegal bid (decreasing bid of doubling partner, redoubling without double).

The result of the flow of data is presented on Figure 2.

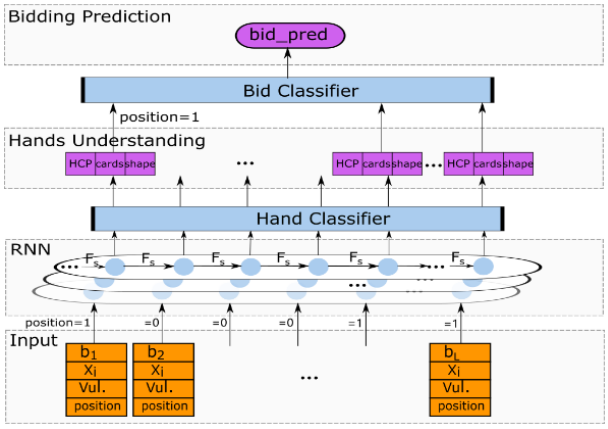

Fig. 2. Bidding system using RNN (source: [19])

Beside proposing bidding model, authors provided visualization system with web user interface and GPU nodes in backend. For human user, it provides access to explanation of bidding system and from the organizational point of view, it allows to resource optimalisation, solution scaling and ensuring availability. Although it is not directly addressing game theory problem, the architecture design surely is important and might have an impact on overall solution.

Agent was deployed with 4 million deals generated by Synrey platform [20] with provided historical bidding sequence and double dummy analysis (70% of data used for training, 20% for validation and 10% for tests). Accuracy of bid prediction made during learning reached 89%. Majority of bidding sequences had length between 8 and 13.

Experiments conducted by researchers showed that their model outperformed popular to compare computer champion Wbridge5 [14] and previously cited work [5].

In conclusion, this work of researchers made another step forward, despite not using reinforcement learning. Authors declared will to continue research using reinforcement learning and self-play techniques to further improve AI ability, in order to achieve superhuman policy in the future.

11. Summary

Contract bridge is considered difficult game and machine learning algorithms have not surpassed humans in solving this game theory problem. However, developers of artificial intelligence make constant steps forward by improving its performance. Approaches presented in this overview paper have used different machine learning methods and have shown different approaches of data representation.

One of the problem without unified solution is representation of bidding history in input of the models. As in [7], we learned that history can be modeled as graph (tree) with separately trained bidding model for every node. This approach however caused large number of learning agents – limiting the depth of the graph. In [5] and [17] authors used ordered list of possible bids, but while the first one used bool array optimized for their model with passive opponents, the latter used representation allowing to map bid to players of both teams and also provide additional states of doubling and redoubling. Finally, an approach in [19] showed possibility to hide history in hidden state of recurring neural network. The last two of mentioned approaches (they are also chronologically the newest) are better suited for addressing the problem in general and may be used more frequently in the future.

Due to rising possibilities and popularity of neural networks, newer solutions used them for developing policy. Even though three solutions based on neural networks were shown, they differ between each other. In [5] authors used self-play to develop clearly artificial policy, model [17] was trained on expert data firstly and improved using reinforcement learning. Lastly presented work [19] used only historic data without reinforcement learning, acknowledging however possibilities of RL pointing it to be direction for future research.

In the future we are likely to see more approaches using reinforcement learning as it is natural way for researchers aiming to create AI able to surpass humans one day.

12. Bibliography

[1] Ameljańczyk A., "Teoria gier", Vol. 690, p. 78, WAT, 1978.

[2] Binmore K., *Game theory: a very short introduction*. OUP Oxford, 2007.

[3] Silver D. and others, "Mastering the game of go without human knowledge", *Nature*, Vol. 550, No. 7676, 354–359 (2017).

[4] Chang M.-S., "Building a fast double-dummy bridge solver", *Technical Report*, NY 1996.
[5] Yeh C.-K., Hsieh C.-Y., Lin H.-T., "Automatic bridge bidding using deep reinforcement learning", in: *IEEE Transaction on Games*, Vol. 10, No. 4, pp. 365–377, 2018.
[6] Gong Q., Jiang Y., Tian Y., "Simple is better: Training an end-to-end contract bridge bidding agent without human knowledge", Real-world Sequential Decision Making, Workshop at ICML 2019, June 14, 2019, Long Beach, USA.
[7] Ho C.-Y., Lin H.-T., "Contract bridge bidding by learning", *Proceedings of the Workshop on Computer Poker and Imperfect Information at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
[8] DeLooze L.L., Downey J., "Bridge bidding with imperfect information", in: *Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games*, pp. 368–373, 2007.
[9] Langford J., Zhang T., "The epoch-greedy algorithm for multi-armed bandits with side information", in: *Advance Neural Information Processing. Systems*, Vol. 20, 1–8, NIPS 2007.
[10] Auer P., Cesa-Bianchi N., Fischer P., "Finite-time analysis of the multiarmed bandit problem", in: *Machine Learning*, Vol. 47, No. 2, pp. 235–256, Springer 2002.
[11] Beygelzimer A., Dani V., Hayes T., Langford J., Zadrozny B., "Error limiting reductions between classification tasks", *Proceedings of the 22nd International Conference on Machine Learning*, pp. 49–56, 2005.
[12] Tu H.-H., Lin H.-T., "One-sided support vector regression for multiclass cost-sensitive classification", *Proceedings of the 27th International Conference on Machine Learning* (*ICML10*), June 21–24, 2010, Haifa, Israel.
[13] Chu W., Li L., Reyzin L., Schapire R., "Contextual bandits with linear payoff functions", *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 208–214, 2011.
[14] Costel Y., *WBridge5*, 2014.
[15] Watkins C.J.C.H., Dayan P., "Q-learning", in: *Machine Learning*, Vol. 8, No. 3, pp. 279–292, Springer, 1992.
[16] Melo F.S., "Convergence of Q-learning: A simple proof", *Institute of Systems and Robotics Tech. Rep.*, pp. 1–4, 2001.
[17] Rong J., Qin T., An B., "Competitive bridge bidding with deep neural networks", *arXiv Prepr. arXiv1903.00900*, 2019.
[18] https://www.bridgebase.com/vugraph_archives/vugraph_archives.php (accessed October 19, 2022).
[19] Zhang X., Liu W., Lou L., Yang F., "AI Enabled Bridge Bidding Supporting Interactive Visualization", *Sensors*, Vol. 22, No. 5, 1877 (2022).
[20] http://www.synrey.com/root.html (accessed October 21, 2022).

Analiza wybranych zastosowań uczenia maszynowego w podejściu do problemu gry brydż

R. JAROSZ

Artykuł przedstawia cztery wybrane podejścia do rozgrywania licytacji w brydżu. W części pierwszej przybliżane są zasady brydża, stanu wiedzy na jego temat oraz krótkie oszacowanie poziomu komplikacji problemu. W części zasadniczej przedstawiono krótkie opisy podejść badaczy do problemu licytacji, badania przedstawione są w kolejności chronologicznej, ukazując ewolucję podejść do problemu. W trakcie opisywania rozwiązań, przybliżane są po krótce matematyczne zasady działania wykorzystanych mechanizmów uczenia maszynowego. Część końcowa podsumowuje przedstawione porównanie rozwiązań i oszacowanie kierunku przyszłego rozwoju.

Słowa kluczowe: brydż, licytacja, niepełna informacja, Q-learning, sieci neuronowe, sztuczna inteligencja, teoria gier, uczenie maszynowe, uczenie ze wzmocnieniem, uczenie z nauczycielem.