24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Bidding in Spades

Gal Cohensius[1] and **Reshef Meir**[2] and **Nadav Oved**[3] and **Roni Stern**[4]

Abstract. We present a Spades bidding algorithm that is superior to recreational human players and to publicly available bots. Like in Bridge, the game of Spades is composed of two independent phases, *bidding* and *playing*. This paper focuses on the bidding algorithm, since this phase holds a precise challenge: based on the input, choose the bid that maximizes the agent's winning probability. Our *Bidding-in-Spades* (BIS) algorithm heuristically determines the bidding strategy by comparing the expected utility of each possible bid. A major challenge is how to estimate these expected utilities. To this end, we propose a set of domain-specific heuristics, and then correct them via machine learning using data from real-world players. The BIS algorithm we present can be attached to any playing algorithm. It beats rule-based bidding bots when all use the same playing component. When combined with a rule-based playing algorithm, it is superior to the average recreational human.

1 Introduction

Spades is a popular card game. Therefore designing strong Spades agents has a commercial value, as millions of games are held daily on mobile applications. Those applications have been downloaded from Google Play store more than any other trick taking game (over 10M times) and produce annual income of several millions of dollars[3, 13].

Spades shares many similarities with games such as Bridge, Skat, Hearts and others that have attracted considerable attention in AI. Recently, AI agents have reached superhuman performance in the partial information game of no-limit poker [4, 5] and to a level of human experts in Bridge, which is considered to be one of the most appraised partial information problems for AI [1, 25, 17]. Spades, however, has received relatively little attention in the literature. Although several Spades bots were made, such as those made by AI-factory (see Related Work), we have no knowledge of strong *publicly available* bots, thus a comparison with the state of the art algorithm is unavailable. Instead, we compared our bidding in Spades (BIS) to humans and to rule-based bidding bots on one of the most popular mobile applications and show that BIS bidding is superior.

The game holds three interesting features from AI perspective: (1) It is a two-versus-two game, meaning that each agent has a partner and two opponents. The partner can be an AI "friend" with a common signal convention or an unknown AI/human where no convention can be assumed; (2) Partly observable state: agents observe their hand but do not know how the remaining cards are distributed between the other players. Each partly observable state at the start of a round can be completed to a full state in $\frac{39!}{13!^3} \cong 8.45 \cdot 10^{16}$ ways; and (3) Goal choosing, as different bids mean that the agent should pursue different goals during the round.

Related work. We first mention two general game-playing algorithms: *Monte-Carlo Tree Search* (MCTS) evaluates moves by simulating many random games and taking the average score [6]. *Upper Confidence bounds applied to Trees* (UCT) is an improved version that biases the sampling in favor of moves that already have higher score [14].

Two groups have made intensive research in the specific area of Spades agents: a group from University of Alberta [18, 19, 20, 21] and AI-Factory group [2, 8, 9, 23]. The latter launched a commercial application called Spades Free. AI-Factory use a knowledge-based bidding module because they found that a "Monte Carlo Tree Search is poor at making bidding decisions" [23]. Whitehouse et al. [23] presented an improved MCTS playing module that beats their strongest heuristic playing agent, they explain that the MCTS must use several heuristics tweaks to be perceived as strong by human players. In a follow-up study, Baier et al. [2] used neural networks in order to emulate human play. They trained the network to predict human's next moves, given a game state. It achieved an accuracy of 67.8% which is an improvement over other techniques such as decision trees.

The Alberta University team considered a simplified version of the game with perfect-information (hands are visible), 3 players, 7 cards, and no partnerships. This reduction was essential in order to reduce the size of possible states of the game, which allow search algorithms to get good results faster. Sturtevant et al. [18] compared the *paranoid* [22] and the max^n [15] algorithms to a handmade heuristic and found that both algorithms were barely able to outperform their handmade heuristic. A followup research from the same group showed that in an n-player game, search algorithms must use *opponents modeling* in order to obtain good results [20]. They proposed the *soft*-max^n algorithm which uses opponent modeling. Interestingly, they used a rule based bidding system, even though the search space is much smaller than classic Spades. In the simple 3-player Spades variant mentioned above, UCT reaches the level of play of prob-max^n [19]. Authors hypothesized that UCT works better in games with high branching factor and low n-ply variance[5] such as Gin-Rummy.

[1] Technion, Israel, email: galcohensius@campus.technion.ac.il
[2] Technion, Israel, email: reshefm@ie.technion.ac.il
[3] Technion, Israel, email: nadavo@campus.technion.ac.il
[4] Ben Gurion University, Israel, email: sternron@post.bgu.ac.il, Palo Alto Research Center (PARC), CA, USA

[5] A measure of how fast the game state can change.

Besides those two groups, dozens of Spades apps are available on Google's play store, most of them have an option of playing with AI players. Currently, The most popular apps are Zinga's Spades Plus,[6] BeachBum's Spades Royale[7] and AI Factory's Spades Free.[8]

Contribution. The main focus of the BIS agent is the decision whether to bid *nil*. BIS uses Monte Carlo simulations combined with heuristic relaxations to obtain heuristic values for nil and non-nil bids. The probability of winning a nil bid is then evaluated using supervised learning from millions of real games.

Combined with a rule-based playing module, BIS is superior to other rule-based bidding algorithms and to the average recreational human, beating humans in 56% of the games. In particular, BIS bids nil more frequently (13.6% of the rounds vs. 12.6%), and still obtains a higher success rate in those rounds (68.8% vs. 63.2% for human players).

Figure 1. A game of Spades. From the agent's perspective, the Left-Hand Opponent (LHO), Right-Hand Opponent (RHO) and the Partner are sitting at West, East and North, respectively.

2 Rules of Spades

Spades is a 4-player trick-taking card game. It resembles Whist, Euchre, Oh Hell, Hearts, Skat, Callbreak and often referred to as a simpler version of Bridge. Its name comes from the rule that the spade suit is strongest (spades are trumps). The game is played in partnerships of two, partners sit across the table from each other and named after the four compass directions: East and West against North and South. The game is played over several *rounds*. At the end of each round both partnerships score points, the winner is the partnership with the highest score that also exceeds a predefined winning goal (usually 500 points). A round begins with dealing 13 cards to each player out of a regular deck of 52 cards. Each round has two phases: *bidding* followed by *playing*.

In the bidding phase, each player, in her turn (passed clockwise), makes a single *bid* (0-13), which states the number of *tricks* she commits to take. The playing phase has 13 tricks, where a trick consists of each player in her turn, playing a single card from her hand onto the table. The player which played the strongest card on the table, wins the trick and will be the leader of the next trick. The first card played in a trick is the leading card and determines the *leading suit* of the trick, other players must follow the leading suit if they can. If in a trick, no spade cards were played, then the highest *leading suit* card is the winner, if a spade card was played then the highest spade card is the winner. The leader cannot lead a spade card unless *spades were broken* or she is *spades tight* (holds nothing but spades). Spades are *broken* when a spade card is played for the first time in the round.

When a round ends, scoring is performed. A partnership that takes at least as many tricks as the combined bid of both partners, receives 10 points times their combined bid, otherwise the partnership loses 10 points times their combined bid. Any extra trick taken beyond their combined bid is called *bag* (or overtrick) and it is worth 1 point. If throughout the game a partnership collects 10 bags, they lose 100 points and 10 bags. Thus players usually aim to take exactly their bid.

For example, assume agent bids 4 and partner bids 2. If their sum of takes is less than 6 tricks then they will lose 60 points, if they will take 9 tricks, then they will receive 63 points. If they had already 288 points (meaning they collected 8 bags during previous rounds), then they will receive 63 however since they cross the 10 bags limit they will lose 10 bags and 100 points, which will results in 241 points.

Bidding 0, also known as *nil*, is a special bid. If a player bids nil then each of the partners in the partnership checks separately whether her bid was successful. A player that bids nil wins 100 points if she individually did not win any trick, and loses 100 points if she did. Terminology can be found in the full version. The complete set of rules can be found at The Pagat website [16].

3 To Nil or Not to Nil, that is The Question

The major decision a player is facing during the bidding phase is whether to bid nil or a regular bid (non nil bid). A nil bid offers a high reward (100 points) but a high risk of being set (−100 points) compared to a regular bid, making the decision a risk-reward trade-off. The major factor of nil bids is shown at Fig. 2. While the score of no-nil rounds is concentrated around 60 points, nil bids result in a risky gamble (see the two peaks of the "Nil" curve).

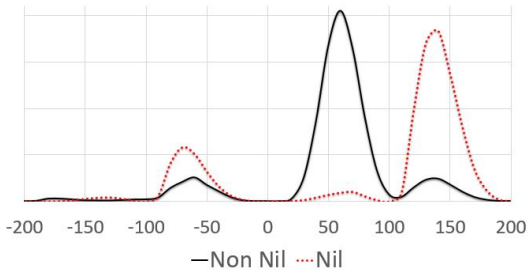

Figure 2. Histograms of the points gained in a round by BIS+Partner, when agent bids either nil or non-nil. The main peak of the "non-nil" histogram (around 60 points) is due to agents fulfilling their regular bid. The left peak (−60) is due to failed bids, and the right peak (130) is due to successful nil bids of the partner.

Deciding to nil does not depend solely on the hand a player holds. The relevant parameters are: (1) the hand's compatibility to a nil bid; (2) the alternative regular bid; (3) the bids by previous players; and (4) the willingness to take risks, which depends on the current scores in the game. For example when the other partnership is about to win the game while having

[6] www.zynga.com/games/spades-plus
[7] www.spadesroyale.com
[8] www.aifactory.co.uk

an easy contract, a desperate nil might be the best option. High alternative regular bid reduces the risk a player should take since the alternative reward is higher. Bidding later in the turn order reveals information about the other players hands. The most important information is the partner's bid. High partner's bid informs that she holds several high cards, which allows the agents to bid nil even with hand containing high cards. A high sum of bids by the three other players reduces the risk for nil since the opponents cannot set the nil without jeopardizing their own bid.

4 The BIS Agent

The BIS agent runs two preprocesses: the first is a Monte-Carlo simulation that estimates the probability that other players hold various cards. The output of these simulations is in the form of Probability Tables, that we denote by PT. The second preprocess uses data from real games played against human players, to estimate success probabilities of nil bids. The output is in the form a real-valued function called *success curves* (SC).

In every round, the bid is determined in the following manner. The agent may use the precomupted data (PT and SC), in addition to the current hand and the sequence of 0-3 previous bids in this round.

Algorithm 1: BIS algorithm

```
1  Input: Hand, PrevBids;
   Result: bid [0-13]
2  regularTakes ←
     CalcRegularTakes(PT,Hand,PrevBids);
3  nilValue ← CalcNilValue(PT,Hand);
4  nilProb ← SC(PrevBids,nilValue);
5  expNilScore ← nilProb ·100 + (1-nilProb) (-100);
6  nilThreshold← CalcNilThreshold(regularTakes);
7  if expNilScore > nilThreshold then
8  |  Return 0;
9  else
10 |  Return regularTakes;
11 end
```

The *regularTakes* variable is an estimation of the number of tricks that the BIS agent can take with high probability. This is a rule-based heuristic estimation based on the current hand and the precomputed probability tables. Details in Sec. 4.1. Similarly, the *nilValue* (line 3) is a heuristic rule-based estimation of the chances to succeed in a nil bid (Sec. 4.2). Since the accuracy of this estimation is critical, we used data from online games to generate a more accurate probability estimation *nilProb*. Generating the Success Curves used in line 4 is the main innovative part of the algorithm. The computation and use of success curves is detailed in Sec. 4.3.

The *nilThreshold* in the deployed version of BIS is a constant value (typically 25 points, but may be higher or lower under endgame conditions that we will later explain). We also consider a more structured way to compute the threshold.

4.1 Regular bid

In a regular bid, BIS tries to estimate the largest number of tricks it can take with high probability. Five features are considered to calculate the regular bid of a hand: (1) side suits high cards (2) spades high cards, (3) long spades suit, (4) short side suits accompanied with spades and (5) the sum of the previous bids. Features 2,3 and 4 are not completely disjoint: the value of a spade card is the maximum between the value it gets from high/long spades and the value it gets from being a potential cut at a short side-suits.

4.1.1 Side Suit High Cards

We use a simplifying assumption that neglects the probability of *finesse*[9]. Thus the first, second and third tricks of each suit will be won by either the A, K, Q, respectively, or by a spade cut. Table 1 presents the probability that in a specific suit, no opponent is void, singleton or doubleton (columns), given the number of cards the agent holds from that suit (rows). These are exactly the probabilities that the agent will take a trick with the A, K and Q respectively (when neglecting the probability of finesse, and having enough blockers). Values in parentheses are worth 0 since the high card does not have enough blockers to be played at the given trick.

For example, in the hand agent holds at Fig. 1, the value of the $K\clubsuit$ is 0.678. This is the probability that both opponents will not be able to cut the second ♣ trick, given the agent holds five ♣ cards (marked with $*$ in Table 1). The value of $Q\diamondsuit$ is 0 (marked with $**$) since agent does not have enough blockers to be played on the third trick.

When an opponent has bid nil or is known to be void at spades, only one opponent may cut so the probabilities are different, and we use a different table instead (see full version).

Table 1. Side suit high card's value. Calculated as the probability that the first/ second/ third trick can not be cut by an opponent, taking into account the player's number of cards from that suit.

cards in side-suit	Probability that both opponents have: > 0 **cards**	> 1 **cards**	> 2 **cards**
0	(0.997)	(0.966)	(0.817)
1	0.994	(0.942) ***	(0.733) **
2	0.990	0.907	(0.624)
3	0.983	0.855	0.489
4	0.970	0.779	0.350
5	0.948	0.678 *	0.212
6	0.915	0.544	0.095
7	0.857	0.381	0.025
8	0.774	0.214	0
9	0.646	0.074	0
10	0.462	0	0
11	0.227	0	0

4.1.2 Spades High Cards

The $A\spadesuit$,$K\spadesuit$,$Q\spadesuit$,$J\spadesuit$ are each worth one trick if they are *mostly protected*. A spade high card is mostly protected if it has more spades than the number of un-owned higher rank spades. This notion comes from the blog *Tactics and Trickery* by Tyler Wong [24]. Formally, the $A\spadesuit$ is worth one trick. The $K\spadesuit$ is worth one trick if the hand contains another spade. The $Q\spadesuit$ ($J\spadesuit$) needs 2 (3) protectors in order to be counted as a take.

[9] A finesse is a method of playing your cards to win a trick with a card lower than your opponents highest card.

4.1.3 Spades Long Suit

Every spade beyond the forth is counted as a take since given the agent holds five spades or more, it is likely no opponent is holding five spades. For example, the spades at Fig. 1 ($A\spadesuit, K\spadesuit, J\spadesuit, 6\spadesuit, 2\spadesuit$) are worth 4 takes: two takes for the $A\spadesuit, K\spadesuit$, the $J\spadesuit$ together with the $2\spadesuit$ as a blocker is worth another take, and another take is due to the fifth $\spadesuit$.

4.1.4 Short Side Suits with Uncounted Spades

BIS's value of short side suits is the probability that it is the only player that can cut in the specific trick, (otherwise, opponents might over cut). Table 1 show this probabilities for void, singleton and doubleton. For example, the cutting value of a Singleton $\diamondsuit$ and two unassigned spades, is the probability that no opponent is having the possibility to overcut on the second and third $\diamondsuit$ tricks, that is $0.942 + 0.733 = 1.675$ (marked with ***,** in Table 1). Each $\spadesuit$ card is counted once where it produce the highest value, either for high/long spades or for cutting short side-suit.

In total, in the example hand displayed at Fig. 1:

- The side suit high cards contribute 0.678 (for $K\clubsuit$) and 0.99 (for $A\heartsuit$). While the $Q\heartsuit, Q\diamondsuit$ contribute 0.
- The $A\spadesuit, K\spadesuit$ contribute 1 take each.
- The $J\spadesuit, 6\spadesuit, 2\spadesuit$ can be counted as a high card ($J\spadesuit$) with a blocker and a fifth $\spadesuit$, and contribute $1 + 1 = 2$.
- Alternatively, the $J\spadesuit, 6\spadesuit, 2\spadesuit$, can be counted as short side suits cuts, which worth $0.942 + 0.733 + 0.624 = 2.3$.

Thus the expected amount of regular takes is $0.678 + 0.99 + 2 + \max\{2, 2.3\} = 5.89$. To this value we add a factor which determined by the sum of previous bids and then it is rounded to the nearest integer to determine the value of regularTakes in line 2 of Alg. 1.

4.2 Nil-Value

BIS heuristiclly estimate the probability of a successful nil bid, we denote this estimation as the hand's nilValue.

Our main relaxation is *almost-suit-independence*:[10] the probability of taking 0 cards from a given suit, depends only on the cards of that suit. Formally:

$$Pr(nil|hand) \approx \prod_{suit \in \{\clubsuit,\diamondsuit,\heartsuit,\spadesuit\}} Pr(nil(suit)|hand \cap suit),$$

This relaxation reduces the number of unknown location of cards from 39 to 13 (minus our holdings from that suit). Thus, enables to efficiently simulate every single suit set of cards in the agent's hand. This relaxation was used in [7, 10] to evaluate nil winning probability in a resembling trick-taking game named Skat.

Therefore, the problem reduces to estimating

$$Pr(nil(suit)|hand \cap suit)$$

in a given suit. Our second simplifying assumption is that in each suit, cards beyond the three lowest are not dangerous.[11]

Monte Carlo deals. As a preprocess, we evaluated the probability of taking zero tricks with each possible set of a single suited cards. The event "$nil(suit)$" depends not only on dealt cards, but also on how players will play, which makes the evaluation ambiguous. Therefore we evaluate a different event $cnil(suit)$ ('cards nil') which only depends on the cards players have. Formally, this event means that "on all tricks of the relevant suit, both opponents can play under one of the agent's cards, and partner cannot cover that card".[12]

We evaluate the probability of the complement event $Pr(\neg cnil(suit)|hand \cap suit)$. There are four cases depending on how many cards are in the suit, and each of them can be written as a union of simple events:

1. In a **void suit** $\emptyset$ the nil can never be set, so $Pr(\neg nil(suit)|hand \cap suit) = 0$ (note that in this case there is no difference between nil and $cnil$).
2. in a **singleton suit** $\{x\}$: both opponents are either void or have at least one card smaller than x, and partner isn't void and has no higher card.
3. in a **doubleton suit** $\{x\overline{x}\}$ is the union of the following two events:
   (a) set at the smallest card x: same as in the singleton suit case.
   (b) set at the second smallest card $\overline{x}$: both opponents are singleton/void or have at least two cards smaller than $\overline{x}$, and partner isn't singleton/void or has no two higher cards than $\overline{x}$.
4. in a **suit with three cards or more**[11] $\{x\overline{x}\overline{\overline{x}}\}$ is the union of the following three events:
   (a) set at the two smallest cards x or $\overline{x}$: same as in the doubleton case.
   (b) set at the third card $\overline{\overline{x}}$: both opponents are doubleton/ singleton/ void or have at least three cards smaller than $\overline{\overline{x}}$, and partner isn't doubleton/ singleton/ void and has no higher card than $\overline{\overline{x}}$.

- evaluating the $\spadesuit$ suit is the same except that partner can not cover by cutting and that a forth spade denials nil (which is a popular heuristics [11]).

To evaluate the simple events used in the above cases, we made a table that contains an entry for each outcome. This is similar to Table 1 but with many more rows, a row for each possible set of cards from the suit. For each entry, we uniformly deal the rest of the suit between the other players 100K times.

For example, the nilValue of the hand in Fig. 1 is calculated as follows:

[10] Suit-independence occurs since players must follow the leading suit if they can, it breaks down when a player is void in the leading suit.

[11] The fourth card in a suit is seldom dangerous since when holding 4 cards from the same suit, only 8% of the deals will allow the opponents to lead a forth trick while the partner can not cover by cutting.

[12] This event leads to failed nil under the following assumptions: (1) no cards from this suit are played on tricks where different suit was lead, (2) both opponents are playing under the agent's card if it is the highest on the table, otherwise they play their highest, (3) partner covers with high card when able, (4) partner can always cut when she is void in the suit.

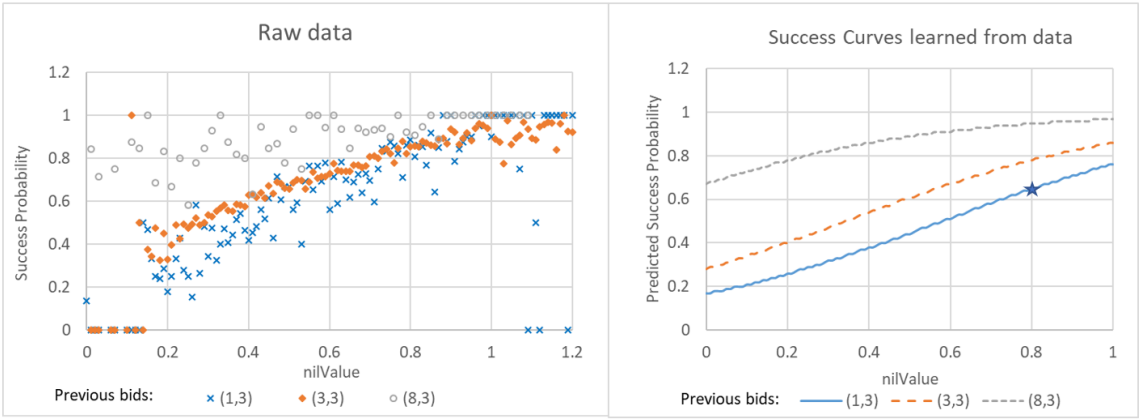

Figure 3. Left: the nilValue and the actual nil success probability of the agent when bidding third, after the RHO bid 3, and the partner bid 1, 3, or 8. Right: the learned success curves for the three bidding sequences on from the left figure.

- ♣ : $K, 9, 5, 4, 3$. The 3 lowest cards are perfectly safe, thus $Pr(cnil(\clubsuit)|hand) = 1$.
- ♢ : $Q\diamondsuit$. The singleton Q has 57.8% to be covered, either when the partner is void, or when the partner holds either the K or the A, or when one of the opponents holds only higher ♢. Thus $Pr(cnil(\diamondsuit)|hand) = 0.578$.
- ♡ : A, Q. According to the respective doubleton table, $Pr(cnil(\heartsuit)|hand) < 0.001$. It is non-zero only due to the chance is that partner is void.
- ♠ : $A, K, J, 6, 2$. The A guarantees a failed nil, thus $Pr(cnil(\spadesuit)|hand) = 0$.
- Thus, the Nil-Value is $1 \cdot 0.578 \cdot 0.001 \cdot 0 = 0$

When the hand contains a void suite, we multiplie the nilValue by a constant factor of 1.15 (hence 'almost-independence'), which is a reasonable approximation that does not require to recompute all the tables.

4.3 Supervised Learning of the Nil Success Probability

The nilValue computed in Sec. 4.2 is providing us with some estimation of the actual probability to succeed in a nil bid. However, we can get a more accurate evaluation of the probability if we take into account the bids of the previous players in this round.

To estimate the nil success rate, we used data from games of the earlier version of BIS that does not use the learning component, combined with the playing module SRP (see Section 5.1). We extracted 2 million rounds from online games played during December 2018 (all games had three bots and a single human player). We only used rounds where the agent bids nil.

For every sequence of previous bids and any nil value, we counted how many nil bids were successful. See Fig. 3(left) for three such sequences. We can see that the nilValue is indeed positively related to the actual success probability, but is not in itself a good approximation, as the probability highly depends on the previous bids. Since BIS never bids nil in some situations (e.g. low nilValue), we also used a noisy variant of BIS for better exploration.

Ideally, we would get a curve of $Pr(nil|nilValue)$ for every possible bid sequence. However, there are $1+14+14^2+14^3 =$ 2955 bidding sequences (0-3 previous bidders, each bid in the range of 0-13). Some of them have enough data to provide a good estimation, but for other bid sequences data is scarce. E.g., there are over 20K rounds for the previous bid sequence "(3,3)", but only about 700 rounds for the sequence "(8,3)", which means no data at all for some nilValues.

To generate the success curves for all bidding sequences, we trained a binary Logistic Regression model on our collected data. The model estimates the nil success probability for each of the 2955 bidding sequences and every possible nilValue.

We then utilize our trained model to retrieve a nil success probability estimate for all possible bidding sequences and nilValues, including for values that do not occur in our data. See Fig. 3(right) for the learned success curves of the three sequences mentioned above.

We generated all 2955 success curves offline, and stored them as the SC tables. The bidding algorithm (see row 4 of Alg. 1) uses the relevant success curve for the actual previous bids, and returns the estimated nil success probability as $SC(PrevBids, nilValue)$. For example, if the agent bids third, previous bids are $(1, 3)$, and the calculated nil value is 0.8 then the estimated nil success probability is 65% (marked with a star in Fig. 3.

Besides Logistic Regression, we experimented with neural networks, random forests and linear regression as well, and got similar results, thus we chose to use Logistic Regression for the following reasons:

- Interpretable - we can easily understand how it weights each feature, as opposed to neural networks.
- Easy to implement and train.
- Explicitly models the probability estimates we are interested in.
- It produces a probability that is monotone in the nil value, as opposed to other methods.

4.4 End-of-game Bidding Modifications

In most rounds, maximizing the expected points in the round is a good approximation to maximizing the winning probability in the game. However when at least one partnership will win the game by fulfilling their contract those two objective differ widely. BIS becomes risk seeking when opponents are

about to win and risk averse when partnership is winning. An example for such modification is the 'complete to 14' bid which means betting on the opponents to fail their bid (as there are only 13 tricks). While this modification may yield negative expected points, it increases the winning probability when opponents are close to winning the game. Those heuristics are detailed at the full version.

5 Experimental Evaluation

In this section, we evaluated our bidding algorithm against other bidding algorithms, and against human players. We then evaluate the impact of the different components of our bidding algorithm.

5.1 Competition with other bots

Setting. We matched BIS with three competing rule-based bidding algorithms (see below). In every game there was one BIS partnership (two agents using BIS for bidding) and one competing partnership. To ensure that only the bidding component is evaluated, all four players used the same playing module (see below).

In each comparison 10K games where played, which is about 35K rounds. Unless stated otherwise, we used a winning goal of 200 points and losing goal of -100 points.[13]

Competing Algorithms:

IO This agent is implemented on `Cardsgame.io` website and uses a fairly simple bidding method. It uses the following nil classifier: if the regular bid ≤ 3 and partner's bid ≥ 4 and hand contains no A or K, and no A-T of spades and no more than 3 spades, then bid nil. It uses the following regular bid: each high spade (A to T) is worth 1 trick, each low spade (9 to 2) is worth 0.4 trick, in side suits, each A is worth 1 trick, each $\{AK\}$ is worth 2 tricks and each $\{Kx\}$ is worth 0.5 a trick. We did not get access to the playing module of this agent.

MS This is an implementation of the bidding scheme denoted as 'simple bidding' which appears in the book *Master Spades* [12]. The instructions in the book for bidding nil (and also the playing instructions) are not concrete enough to write them down as an algorithm. We therefore combined the bidding algorithm with the naïve nil classifier of RB. The regular bid is the following: Nine cards are worth one trick each: Aces, non-singleton Kings and the $Q\spadesuit$ (if it is not a singleton or doubleton that does not contain the $A\spadesuit$). Each $\spadesuit$ beyond the first three is worth a trick while a void or a singleton spade reduce the bid by one. A side-suit void or singleton, together with exactly three spades increases the bid by one.

RB Rule-based Bidder. This is the previous bidder that was implemented in the application we use for evaluation. Its regular bid calculated as a sum of values of each card in hand. Each card has a value that depends on the number of cards from that suit in the hand. The naïve nil classifier bids nil if the lowest, second lowest and third lowest cards from each suit are not larger than $5, 8$ and 10 thresholds respectably.

[13] The (+200/-100) goal was a common setting on the application at the time. We received similar results with other goals.

Playing modules:

WRP A Weak Rule-based Player. Strength of an average recreational human player. When combined with the RB bidding module, wins almost 50% of the games when plays vs. recreational human players.

SRP A Strong Rule-based Player - When combined with the RB bidding module, wins 56% of games vs. RB+WRP; and 54% of the games vs. recreational human players.

UCT An implementation of the UCT algorithm [14] for the game of Spades. We use a time limit of 3 seconds to decide the number of samples. Further details about our implementation and the limitations of UCT are found in the full version.

We can also use the general UCT algorithm for bidding and not just for playing. We denote by UCT (X) the SRP playing module where the last X tricks are replaced with UCT.

Results. Table 2 shows that for three different playing modules, the bidding module of BIS is stronger than the other bidding modules.

Table 2. Comparison of BIS bidding to other bidding algorithms. In each comparison, the playing module is fixed for both bidding modules. The comparison is in two aspects: win ratio and average points per round.

Opponents' Bidding	**Playing Module**	BIS's win rate	**Average points BIS : opponents**
RB	WRP	51.9%	**53.2** : 50.0
MS	WRP	66.9%	**50.1** : 38.2
IO	WRP	68.6%	**52.8** : 46.5
RB	SRP	52.3%	**56.2** : 52.8
MS	SRP	67.7%	**50.8** : 38.4
IO	SRP	67.1%	**51.6** : 46.3
RB	UCT (3)	52.1%	**52.0** : 48.5
RB	UCT (5)	52.9%	**48.5** : 40.3

Our results shows that the UCT playing module is stronger than SRP only when activated at the last several tricks, this is because of the three seconds time limit which does not allow the UCT to search accurately when activated earlier. If we would like to use UCT as a bidding module, it would have to search a huge space. We are currently looking into how UCT can be combined with our BIS agent without using excessive computation time.

5.2 Playing against humans

Setting and dataset. Our BIS agent was deployed in a popular mobile Spades application.[14] We extracted more than 400K rounds from games played during October 2019, between a partnership of two BIS+SRP agents and a partnership composed of a BIS+SRP and a human player.

Results. The overall winning rate of the BIS partnership is 56% of all games, which are 2 percentage points above the performance of the previous rule-based partnership used by the platform (RB bidding module with the same SRP playing

[14] We disclose the name of the platform in a note to the reviewers.

module). Note that this data suffers from selection bias because games were only recorded if they ended, while humans tend to quit games when they are losing badly.

To better understand the strenghths and weaknesses of BIS we divided all rounds to types according to the bids made by players.

Table 3 shows that in almost all round types, the BIS+SRP partnership obtains a higher score. This is true also when partitioning no-nil rounds according to the sum of bids (Fig. 4). The biggest points gaps in favor of BIS+SRP are in 'double nil' rounds (both partners bid nil) and when the total bids exceed 14. We conjecture that those rounds are a result of bidding blunders made by humans.[15]

One exception where the BIS+H partnership scores higher is when bids sum up to exactly 14. This is explained by BIS end-of-game bidding modifications (see Sec. 4.4). The other exception is when the human's partner bids Nil, demonstrating that the poor performance of the partnership is due to the human part.

Table 3. Comparison between a partnership of two BIS+SRP against BIS+SRP and a human. The comparison is broken down by round types. The BIS+SRP partnership is generating more points in rounds containing nils.

nil position	# rounds	Average points			Successful nils		
		BIS+BIS	:	H+BIS	BIS+BIS	:	H+BIS
BIS+BIS	117782	57.3	:	47.7	68.8%	:	-
H	73157	50.7	:	49.2	-	:	63.9%
Partner of H	57392	51.9	:	62.2	-	:	70.0%
Nil vs. Nil	102132	67.4	:	60.5	44.2%	:	39.6%
Double Nil	691	64.5	:	-29.3	-	:	32.4%
No Nil	95928	48.9	:	43.6	-	:	-

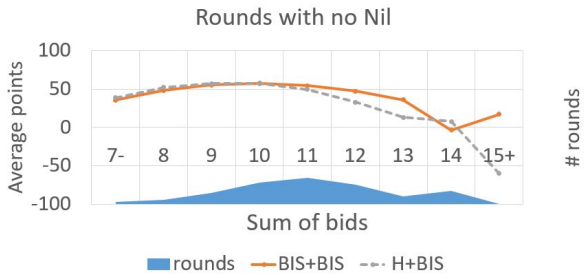

Figure 4. Average points per round in rounds where no player has bid nil, broken down based on the sum of the four bids.

5.3 Impact of BIS components

This section aims to reveal the significance of several components in BIS bidding by comparing a BIS partnership to a partnership that has the specific component disabled or modified. In all simulations, all four bots use the SRP player.

5.3.1 Success Curves

A simple variant of the BIS bidding algorithm would use a single Success Curve, without taking the previous bids into account. The current BIS algorithm beats the simple variant in 51.2% of the games, and gains 2.5% higher score on average. This means that the success curves are responsible for about 1/4 of the overall improvement that BIS obtains over the best rule-based bidder RB.

[15] Double nil bid is rarely beneficial (BIS never bids nil if its partner bid nil). Sum of bids higher than 14 is usually an indication that a human player is vastly overvaluing their hand, since BIS bids conservatively.

Table 4. BIS Vs BIS with a single component disabled

Opponents	BIS's win rate	Average points BIS : opponents
Single success curve	51.2%	44.7 : 43.6
No end-game conditions, winning goal of 200	52%	48.3 : 49.9
No end-game conditions, winning goal of 1	52.8%	43.5 : 52.9

5.3.2 End-of-game Bidding Modifications

The end game conditions are a set of heuristics described at Subsection 4.4. BIS with the end game conditions won 52% of the games under the usual winning goal of 200 points. As expected, BIS with the endgame module obtains *fewer* points per round. When the winning goal was set to 1 point (single round games), the win rate increased while the average points decreased.

5.3.3 Bid-sensitive Nil Threshold

The BIS algorithm uses a threshold as a cutoff point to decide whether to bid nil (see Lines 6-7 in Alg. 1). The current algorithm uses a constant threshold (specifically, 25) that is based on the expected score of a non-nil bid. It may seem wiser to use more available information to determine this threshold.

Indeed, we implemented a variation of the BIS algorithm (BIS*), which tries to evaluate the expected score *of the partnership* once if the agent would bid nil and second if she will bid a non-nil bid, using the regularTakes value and the bid of the partner, if known.

When playing against each other, BIS* was slightly worse. One possible explanation is that an inaccurate estimation is worse than using a constant threshold. We hope to better understand the weak points of the estimation and improve the threshold decision in future work.

6 Conclusion

This work is the first to publish a Spades bidding algorithm that outperforms recreational humans. Our hope is that it will serve as a baseline for future work which will allow other teams to build stronger Spades bidding modules.

BIS is flexible in the sense that it can bid in several variations of Spades, such as Cutthroat Spades (i.e. Solo), Mirror, Spades with jokers, Whiz and Suicide, each of them require very little game-specific modifications. We conjecture that the methods we used might produce strong bidding modules in other trick-taking games such as Skat, Whist and Callbreak. In the first two games, a nil classifier is a major part of the bidding and our nil classifier, with slight modifications can be used. Our regular bid evaluator needs slight modification to the value of cards in order to be used in those other games.

The main takeaway message that goes beyond applications to trick taking card games, is our approach of combining rule-based heuristics and learning. That is, we first generate a rule-

based heuristics (in this case, of the probability to succeed in a nil bid), and then apply machine learning on past data, using this heuristic as the main feature, to get an improved estimation.

Future research is focused on better tackling the weak points of the bidding module (cases where probability estimations are off), and on improving the playing module. Our goal is to develop a combined Spades agent with super-human strength.

ACKNOWLEDGEMENTS

We would like to express our gratitude to Nathan Sturtevant and Stephen Smith for their experts advice, many thanks to Einar Egilsson for granting us access to his bidding algorithm.[16] We would like to thank an anonymous game studio that gave us access to their data of completed games which contribute to our evaluation of BIS performance.

REFERENCES

[1] Asaf Amit and Shaul Markovitch, 'Learning to bid in bridge', *Machine Learning*, **63**(3), 287–327, (2006).
[2] Hendrik Baier, Adam Sattaur, Edward Powley, Sam Devlin, Jeff Rollason, and Peter Cowling, 'Emulating human play in a leading mobile card game', *IEEE Transactions on Games*, (2018).
[3] Mehul Boricha. 10 best card games for android in 2020. `https://www.techrrival.com/best-card-games-android/`.
[4] Noam Brown and Tuomas Sandholm, 'Superhuman ai for heads-up no-limit poker: Libratus beats top professionals', *Science*, **359**(6374), 418–424, (2018).
[5] Noam Brown and Tuomas Sandholm, 'Superhuman ai for multiplayer poker', *Science*, **365**(6456), 885–890, (2019).
[6] Bernd Brügmann, 'Monte carlo go', Technical report, Citeseer, (1993).
[7] Michael Buro, Jeffrey Richard Long, Timothy Furtak, and Nathan Sturtevant, 'Improving state evaluation, inference, and search in trick-based card games', in *Twenty-First International Joint Conference on Artificial Intelligence*, (2009).
[8] Peter I Cowling, Edward J Powley, and Daniel Whitehouse, 'Information set monte carlo tree search', *IEEE Transactions on Computational Intelligence and AI in Games*, **4**(2), 120–143, (2012).
[9] Sam Devlin, Anastasija Anspoka, Nick Sephton, Peter I Cowling, and Jeff Rollason, 'Combining gameplay data with monte carlo tree search to emulate human play', in *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, (2016).
[10] Stefan Edelkamp, 'Challenging human supremacy in skat–guided and complete and-or belief-space tree search for solving the nullspiel', in *31. Workshop" Planen, Scheduling und Konfigurieren, Entwerfen*, (2018).
[11] Einar Egilsson. cardgamesio. `cardgames.io/spades/`.
[12] Steve Fleishman, *Master Spades - Advanced Card Playing Technique and Strategy at Spades*, Master Spades, 2002.
[13] Joe Hindy. 15 best card games for android! `https://www.androidauthority.com/best-android-card-games-581095/`.
[14] Levente Kocsis and Csaba Szepesvári, 'Bandit based monte-carlo planning', in *European conference on machine learning*, pp. 282–293. Springer, (2006).
[15] Carol Luckhart and Keki B Irani, 'An algorithmic solution of n-person games.', in *AAAI*, volume 86, pp. 158–162, (1986).
[16] John McLeod. Paget - spades rules. `www.pagat.com/auctionwhist/spades.html`. Accessed: 2019-06-05.
[17] Jiang Rong, Tao Qin, and Bo An, 'Competitive bridge bidding with deep neural networks', in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 16–24. International Foundation for Autonomous Agents and Multiagent Systems, (2019).
[18] Nathan Sturtevant, 'A comparison of algorithms for multiplayer games', in *International Conference on Computers and Games*, pp. 108–122. Springer, (2002).
[19] Nathan Sturtevant, 'An analysis of uct in multi-player games', *ICGA Journal*, **31**(4), 195–208, (2008).
[20] Nathan Sturtevant and Michael Bowling, 'Robust game play against unknown opponents', in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 713–719. ACM, (2006).
[21] Nathan Sturtevant, Martin Zinkevich, and Michael Bowling, 'Prob-max^n: Playing n-player games with opponent models', in *AAAI*, volume 6, pp. 1057–1063, (2006).
[22] Nathan R Sturtevant and Richard E Korf, 'On pruning techniques for multi-player games', *AAAI/IAAI*, **49**, 201–207, (2000).
[23] Daniel Whitehouse, Peter I Cowling, Edward Jack Powley, and Jeff Rollason, 'Integrating monte carlo tree search with knowledge-based methods to create engaging play in a commercial mobile game.', in *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*, (2013).
[24] Tyler Wong. Spades: tactics and trickery. `www.sharkfeeder.com/spadsbook/basicbidding.html`.
[25] Chih-Kuan Yeh, Cheng-Yu Hsieh, and Hsuan-Tien Lin, 'Automatic bridge bidding using deep reinforcement learning', *IEEE Transactions on Games*, **10**(4), 365–377, (2018).

[16] Available to play at cardgames.io/spades/