The Game of Bridge: A Challenge for ILP

Swann Legras[1], Céline Rouveirol[2(B)], and Véronique Ventos[1,3(B)]

[1] NUKKAI Inc., Paris, France
vventos@nukk.ai
[2] L.I.P.N, UMR-CNRS 7030, Univ. Paris 13, Villetaneuse, France
celine.rouveirol@lipn.univ-paris13.fr
[3] L.R.I., UMR-CNRS 8623, Univ. Paris-Saclay, Orsay, France

Abstract. Designs of champion-level systems dedicated to a game have been considered as milestones for Artificial Intelligence. Such a success has not yet happened for the game of Bridge because (i) Bridge is a partially observable game (ii) a Bridge player must be able to explain at some point the meaning of his actions to his opponents. This paper presents a simple supervised learning problem in Bridge: given a 'limit hand', should a player bid or not, only considering his hand and the context of his decision. We describe this problem and some of its candidate modelisations. We then experiment state of the art propositional machine learning and ILP systems on this problem. Results of these preliminary experiments show that ILP systems are competitive or even outperform propositional Machine Learning systems. ILP systems are moreover able to build explicit models that have been validated by expert Bridge players.

1 Introduction

Designs of champion-level systems dedicated to a game have long been considered as milestones for Artificial Intelligence; IBM computer Chess Deep Blue beating in 1997 world champion Gary Kasparov was one of these adventures, another more recent one being DeepMind Alphago outperforming in 2017 World Champion Ke Jie at the game of Go [15]. One major challenge still remaining concerns the game of Bridge where the level of robots is far from the best human players. There are two main reasons for that: (i) Bridge is a partially observable game (ii) a Bridge player must be able to explain at some point the meaning of his actions. For example, an opponent can ask for explanations on the inferences related to the choice of an auction rather than another. The need for explainability coupled with the fact that knowledge in Bridge is relational let us think that Bridge is a killer application for ILP. As a proof of concept, this paper[1] presents a simple supervised learning problem in Bridge: given a 'limit hand', should a player bid or not, only considering his hand and the context

[1] This research work is conducted as part of the νBridge project (former name AlphaBridge) supported by NukkAI Inc., Paris.

F. Riguzzi et al. (Eds.): ILP 2018, LNAI 11105, pp. 72–87, 2018.
https://doi.org/10.1007/978-3-319-99960-9_5

of his decision. After describing this problem and how it has been modeled, we experiment state of the art propositional machine learning and ILP systems on this problem. Results of these preliminary experiments show that ILP systems are competitive or even outperform propositional Machine Learning systems on this problem. More importantly, ILP systems are flexible enough, given their explicit background knowledge and the fact that they build explicit and expressive models, to support the expert Bridge players in the modelling process.

The goal of the paper is to demonstrate that Bridge displays a certain number of characteristics that would benefit from a relational representation. Building explicit rules or patterns for Bridge indeed requires the expressiveness of ILP: given his thirteen cards (his hand), a player makes decisions according to properties of the hand. These properties can be related to specific cards, sets of cards or other abstractions of the hand. The symbolic approach is quite flexible, and allows experimenting with various abstractions of examples description through the use of background knowledge, in close interactions with the experts. The relational nature of the language used for describing the background knowledge facilitated the task of the experts.

The plan of the paper is as follows: we first describe the target Bridge problem in Sect. 2. We then model this problem as a binary classification one and design a first propositional representation of examples to be handled by a state of the art propositional learning system (Sect. 3). Section 4 describes alternative relational representations of examples as well as settings for the two state of the art ILP systems studied, Aleph and Tilde. Section 5 gives some empirical comparisons of the models learnt in those quite different contexts (accuracy, complexity). Relational models are then discussed and assessed by Bridge experts in Sect. 6. Finally, Sect. 7 draws some conclusions and some perspectives for the νBridge project.

2 Description of the Bridge Problem

In the present section, we give a short overview of the game of Bridge and we present the decision problem handled in this paper. The interested reader can refer for instance to [12] for a more complete presentation.

Bridge is a trick-taking card game opposing four players divided in two partnerships (pairs). A standard 52 card pack is shuffled and each player receives a hand that is only visible to him. Pairs stand across each other. A Bridge deal is divided into two major playing phases using different devices (represented Fig. 1): the bidding phase and the card play. The goal of the bidding is to reach a contract which determines the minimum number of tricks the pair commits to win (between seven and thirteen) during the card play, either with no trump or in a determined suit as trump. The different final contracts are denoted by nS where n is a number between 1 and 7 and S $\in$ (♠, ♡, ♢, ♣, NT). The contract nS determines the minimum number of tricks the pair commits to win (n + 6) and which suit is the trump, NT to expressing the fact that there is no trump. For instance 4♡ is fulfilled if the number of tricks won is at least 10 (4 + 6)

Fig. 1. The two devices used in Bridge: bidding cards (left) and standard cards (right)

with a ♡ trump. The major problem during the bidding phase is to be able to evaluate the trick-taking potential of a pair of hands, each player not knowing what his partner holds. Players use bidding cards to pass information to their partner about their hand. The last bid represents the contract to be played. During the card play, the goal is to fulfill (or to defeat for the defending side) the contract reached during the first phase. Each player plays a card at his turn. As this paper focuses on a bidding problem we will not go into further detail on this phase.

2.1 Opening Bid Problem

During the bidding phase, the dealer is the first to bid and has to decide between opening (the bidding) or passing. If he passes then the opponent at his left has to decide between opening and passing and so on. It can happen, although rarely, that all 4 players pass: the deal is a 'passout'.

When deciding whether to open or not, the player evaluates his hand by counting the total of 'High-Card Points' (HCP: Ace = 4, King = 3, Queen = 2, Jack = 1). It is a way to rapidly assess the potential of a hand as higher card have more chance to earn a trick during card play. Generally, early bids like opening bids are made according to as set of rules. For instance, in modern Standard American system (SAYC), 1-of-a-suit opening requires at least 12 HCP. In some limit cases, experts allow themselves to deviate slightly from the rules and produce a bid that does not meet with the generally admitted conditions of application of a rule. Thus 11HCP hands are limit cases in which some players may decide to open and some others not. The decision to open or not an 11HCP hand is called in the following the opening bid problem. The opening bid problem is linked to a Bridge situation as that represented in Fig. 2. The example (refered to as Example 1) of this figure will be used in the following to illustrate our explanations.

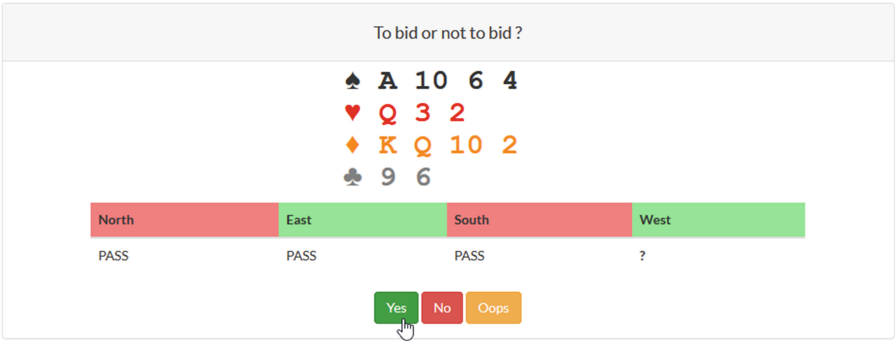

Fig. 2. Example 1: a bidding decision for West (Color figure online)

Parameters: when holding an 11HCP hand, the decision to open or not is based on several parameters: the position (1st if he is the dealer, 2nd after one Pass, 3rd after two Pass or 4th after 3 Pass), the vulnerability measuring the risk/reward score-wise and the evaluation of the hand.

Players are represented by *North*, *East*, *South* and *West* (abbreviated to N, E, S, W) the pairs being NS and EW. Here, *North* is the dealer. The vulnerability of the two sides is represented by the color of the players (green is Not vulnerable and red Vulnerable). The decision is related to *West* according to his hand, his position (4th since there are 3 passes from the other players), and his vulnerability (Not Vulnerable since his color is green).

By clicking on 'Yes', *West* decides to open and then he deviates slightly from the 12 HCP rule since his hand has only 11 HCP (4+2+3+2).

The ability to wisely depart from the rule framework does make a difference at an early stage of the bidding phase because the final contract (and therefore the score) depends heavily on initial actions.

Finally, it is said that in these situations, Bridge players exercise their hand judgment without being able to elicit their decision making process: this skill is built on strong and personal experience and is very difficult to get. The decision and evaluation also vary a lot from one player to another, even of similar level.

3 Learning Problem and First Model

The opening bid problem is a binary classification problem where Task T consists in predicting if a given expert opens or passes according to a Bridge situation like that in *Example 1*. The input of the learning problem is a set of n labeled examples $(x_i, class_i)$ where $class_i$ belongs to $\{+, -\}$ (positive or negative examples), the output is a classifier, i.e., a function $h(x)$ that assigns each example x to its class: $+$ (open) or $-$ (pass).

3.1 DataSets

We randomly generated 6 sets of unlabeled samples. Those sets have been labeled by four Bridge experts $(E_1, \ldots, E_4)$ who are among the best 100 players of their home country. These experts have similar level but different styles that affect their decision to open or not (with the exact same cards, 2 experts are likely to make different decisions). Table 1 sums up the set features and some statistics about the labeling of examples.

Table 1. Samples sets

Labeled set	S_1	S_2	S_3	S_4	$S_{5,1}$	$S_{5,2}$	S_6
Unlabeled set	R_1	R_2	R_3	R_4	R_5	R_5	R_5
Expert	E_1	E_2	E_2	E_3	E_4	E_4	E_4
Size	1000	1000	970	790	1222	1222	1079
Pos./Neg	768/232	681/319	540/430	603/187	681/541	582/640	560/519
Pos rate (%)	76.80	68.10	55.67	76.33	55.72	47.63	51.90

Sets R_i, $i \in \{1..4\}$ were tagged once and by only one expert. Let us call S_i, $i \in \{1..4\}$ the corresponding sets of labeled examples.

The opening bid decision is a complex issue, being not unusual that the same expert makes different decisions while facing the exact same situation. To get an idea of expert's consistency, expert E_4 was given twice the set R_5 to label, a number of months apart and with the deals arranged in a different order. The 2 labeled sets are called $S_{5,1}$ and $S_{5,2}$. Finally, $S_6 = S_{5,1} \cap S_{5,2}$ represents the set of samples of R_5 which have been 'consistently' labeled by E_4. The consistency rate of expert E_4 when labeling R_5 is only 88.30%.

Bridge players with similar level may have very different features. For instance E_4 is a conservative player who seldom trespasses the limits fixed by the bidding system while E_1 and E_3 are 'aggressive' players. Their ratio of positive examples in the last row of Table 1 is consistent with their presumed aggressiveness.

3.2 Propositional Representation of Examples

We first represent examples using a propositional representation.

Definition 1 (Propositional representation of examples). *A labeled example is represented by: [c_n with $n \in \{1..52\}$, position, vulnerability, class]. We use the following ascending order for cards: ♣, ♢, ♡, ♠ and for each suit : 2, ..., 10, J, Q, K, A.*

c_i = 1 if the hand has the corresponding card and 0 otherwise, position $\in \{1..4\}$, vulnerability $\in \{1..4\}$ with 1 = the two pairs are not vulnerable, 2 = not vulnerable opponents vulnerable, 3 = vulnerable opponents not vulnerable and 4 = the two pairs are vulnerable, Class = 1 if the expert opens and 0 if he passes.

3.3 Binary Classification Using SVM

We chose Support Vector Machines among various paradigms that are used for learning binary propositional classifiers (e.g. Decision Trees, Bayesian Classification, Perceptron, etc). Experiments were performed with a SVM learner with linear kernel as implemented in the Scikit-learn[2] toolbox [14].

4 Building Rule Sets with ILP Systems

The ILP systems used in the experiments are Aleph [16] and Tilde [3]. They are both state of the art ILP system, quite different in their learning process. Aleph learns from entailment. Given a set of positive examples E^+ and a set of negative examples E^- and a background theory B, the learning goal is to find a hypothesis H in the form of a Prolog program, such that $\forall e \in E^+ : H \wedge B \models e$ and $\forall e \in E^- : H \wedge B \not\models e$. Examples in this framework are single literals representing the class ($open/1$). Background knowledge is represented as a mixture of facts and rules that can be used to enrich the examples' description.

Aleph's learning proceeds top-down. It learns sets of rules using a covering approach. Aleph selects an example e not covered by the current theory. A learning loop starts by building a bottom clause from e, a most specific clause e given e and B). This bottom clause will be the lower bound of Aleph's search space. Aleph then explores top-down generalizations of bottom clause that satisfy the language bias and mode declarations. Several search strategies have been implemented in Aleph, we have used the default one (best first).

Tilde learns a relational decision tree from interpretations. Given a set of classes C (each class label c is a nullary predicate), a set of examples E (each element of E is of the form (e, c) with e a set of facts (interpretations) and c a class label) and a background theory B, the goal is to find a hypothesis H (a Prolog program), such that $\forall (e, c) \in E, H \wedge e \wedge B \models c$, and $\forall c' \in C - c : H \wedge e \wedge B \not\models c'$. As quoted in [3]:

> *Since in learning from interpretations, the class of an example is assumed to be independent of other examples, this setting is less powerful than the standard ILP setting (e.g., for what concerns recursion). With this loss of power comes a gain in efficiency, through local coverage tests. The interesting point is that the full power of standard ILP is not used for most practical applications, and learning from interpretations usually turns out to be sufficient for practical applications.*

This is the case for our classification problem.

A first order logical decision tree (FOLDT) is a binary decision tree in which the nodes of the tree contain a conjunction of literals and different nodes may share variables, under the following restriction: a variable that is introduced in a node must not occur in the right branch of that node. In order to refine a node with associated query Q, Tilde computes the refinement of Q given the language

[2] http://scikit-learn.org/stable/.

bias and chooses the refinement that results in the best split. The best split is the one that maximizes a certain quality criterion by default the information gain ratio. The conjunction at a given node consists of the literals that have been added to Q in order to produce its refinement.

The ILP systems allow for using a powerful and flexible target representation language through the use of explicit background knowledge, here represented as a set of definite clauses. To cope with the increased size of the hypothesis space, both Aleph and Tilde allow the user to define the search space through sophisticated *language bias*. This language bias can be seen as a grammar-like user defined specification of the search space.

4.1 Relational Representation

In this section, we present the different background knowledges (denoted BK_i in the remainder of the paper) that have been used as an input of the ILP systems as well as modeling assumptions used to obtain them. These BK stem from a joint work with the Bridge experts in order to achieve both an acceptable Bridge-wise representation and an acceptable learning performance. These objectives will be assessed in Sect. 5, the expert validation of the results is given in Sect. 6.

The first BK BK_0 is a simple translation of the propositional representation described in Sect. 3.2.

BK_0: The following predicates form the common background for all BKs (E represents the example primary key).

- $open(E, Class)$ with $Class = pos$ or neg
- $has\_hand(E, H)$ where H is a list of card constants representing the hand's example
- $has\_card(E, C)$ is $true$ if C is a card occurring in E's hand
- $position(E, P)$ with P a digit between 1 and 4 representing the position of the expert
- $vuln(E, V1, V2)$ with $V1$ and $V2 = g$ (not vulnerable) or r (vulnerable) representing the vulnerability of the player and of its opponents.

According to BK_0, *Example 1* in Fig. 2 (denoted by $e1$ in the following), is represented by the following facts:
$open(e1, pos).$ $has\_hand(e1, [c6, c9, d2, d10, dq, dk, h2, h3, hq, s4, s6, s10, sa]).$
$position(e1, 4).$ $vuln(e1, g, r).$
$has\_card(e1, c6).$ $has\_card(e1, c9).$ ... $has\_card(e1, sa).$

BK_1: In order to introduce some abstraction and Bridge knowledge in the hand's description, we added some predicates to characterize card properties according to a Bridge expertise such as: the suit, the rank, the suit category (minor/major) and the rank category of the card ($small\_card$ from 2 to 10 or $honor$ from Jack to Ace), represented with the following predicates:

$has\_suit(Card, Suit).$ $has\_rank(Card, Rank).$
$honor(Card).$ $minor(Card).$ $small\_card(Card).$

Expert's validation related to the intermediate BK obtained (see Sect. 6), allows us to introduce higher level predicates related to hands' properties:

$nb(E, Suit, Number)$. $plusvalue(E)$.
$lteq(Number, Number)$. $gteq(Number, Number)$.

nb describes the length of a particular suit. nb($h1$, heart, 3) expresses the fact that the hand in *Example 1* has 3 cards in ♡. *plusvalue* allows to rate a hand as 'good'.

One of the difficulties to adequately represent Bridge expertise was to determine at which granularity level the example had to be represented. We illustrate this point with the concept of exact distribution (number of cards per suit) related to a hand, that is an important parameter to evaluate a hand's potential outside the value of HCP. As an illustration, the exact distribution of the hand of *Example 1* is 4-3-4-2 (4♠, 3♡, 4♢ and 2♣). However, a Bridge player will often reason in terms of hand patterns denoting the distribution of the thirteen cards by decreasing order irrespective of the suits in question (e.g. 4-4-3-2 for *Example 1*). The term distribution will further refer to this notion and not to the exact distribution. This notion can be abstracted slightly more by classifying hands into three main classes: *balanced*, *semi_balanced* and *unbalanced* according to their distribution. For example, balanced hands are hands with no short suit (0 or 1 card). The following rules allow to saturate examples with this property:

```
balanced(E) :- distribution(E,[4,3,3,3]).
balanced(E) :- distribution(E,[4,4,3,2]).
balanced(E) :- distribution(E,[5,3,3,2]).
```

Thus, we can change the granularity of the hypothesis language: just give the exact distribution, replace the exact distribution by the distribution, remove both and only use the predicates *balanced*, *semi_balanced* and *unbalanced*, etc.

BK_2: Finally, $BK_2 = BK_1 \cup list\_honor(E, Suit, ListH)$ where $ListH$ is the list of honors of suit $Suit$ in the hand. $list\_honor/3$ introduces an abstraction of the list of cards of a given suit since only the most important cards (honours) are taken into account. We will check in the experiments if this additional predicate impacts and hopefully improves the quality of learned models.

4.2 Aleph and Tilde Settings

In this section, we discuss the different settings used in Aleph and Tilde in the experiments shown in next section. Some of them are inherent to the algorithm used, some are choices to either increase performance, readability of the output or simplicity of implementation. We only detail the most important settings for each system. We refer to the user guides of both systems [2,16].

Target Predicate. The target predicate for Tilde is $open(E, Class)$, and $open(E)$ for Aleph.

Language Bias. After background predicates have been defined, the language bias describes more precisely how these predicates should be used to form rules. The most important information for Aleph is the definition of predicate modes (should constants or variables – input or output[3] – occur in which arguments of the literals. Preliminary tests have shown that the predicate *distribution* better behaves with Aleph than with Tilde.

The lookahead capability of Tilde allows creating nodes labeled with conjunctions of $nb(E, Suit, Value)$ and *gteq* or *lteq* describing length of specific or major/minor suits. Tilde is thus able to generalize the idea of distribution by performing multiple tests on the different length of suits with the predicate. Aleph handles less easily this kind of abstraction, that requires to learn longer clauses, with the counterpart of explosion of search complexity, but behaves well with the *distribution* predicate. We therefore chose to disallow Tilde to use *distribution* predicate in its language bias.

Negative Literals. Aleph does not produce rules that have negative literals in their body (it only builds definite clause programs). We have therefore encoded in the Background Knowledge some predicates representing the negation of another predicate (e.g *notplusvalue*). Tilde as a decision tree learner obviously does not need such explicit negative literals.

Search Options. The goal of this paper being to apply state of the art techniques on a new problem, we chose to use default search options whenever possible in both Aleph and Tilde. Tilde's search options were all default ones (for Aleph: *mincase* = 2, *minacc* = 0; for Tilde: *heuristic* = gainratio, *pruning* = c45 safe pruning, *stopping criterion* = mincase).

When using Aleph, the only options that deviated from default values were *clauselength* (the maximum length of rules generated) that was set to 6, *minpos* (the minimum positive coverage of each rule) set to 2 and *noise* (the maximum number of negative examples a rule can cover) was set to 5. We set this last value empirically, by running the problem with various BKs, datasets and noise values. We observed that a noise of 0 was not appropriate, as even experts may contradict themselves when labeling examples for this problem (see Sect. 3.1). The mean performance curves across datasets do not show any clear trend to help us setting this noise parameter (see Fig. 9 in the Appendix[4]). We have therefore chosen 5 as a good compromise between the performance and specificity of rules obtained. The parameter *minacc* was set to its default value (0) as we observed a loss of accuracy when setting it over 0.5.

[3] An input variable already occurs in the left part of the clause under development, while an output variable does not occur in the current clause (and in its head) and is therefore existentially quantified.

[4] http://www.nukk.ai/ILP2018/.

5 Experiments

We have run experiments for all datasets and systems. Because of lack of space, we only comment here on graphs for datasets S_1 et $S_{5.2}$ labeled by experts with different profiles: E_1 which is very structured and concise, and E_4 whose decision making process is driven by special cases. These assumptions will be confirmed in Sect. 6. Graphs for other datasets are provided in the appendix (Fig. 7). Their complete analysis is outside the scope of the paper.

The performance measure to assess all classifiers is accuracy: $\frac{TP+TN}{P+N}$ where TP and TN denote the true positive and true negative examples, whereas P and N denote the number of positive and negative examples.

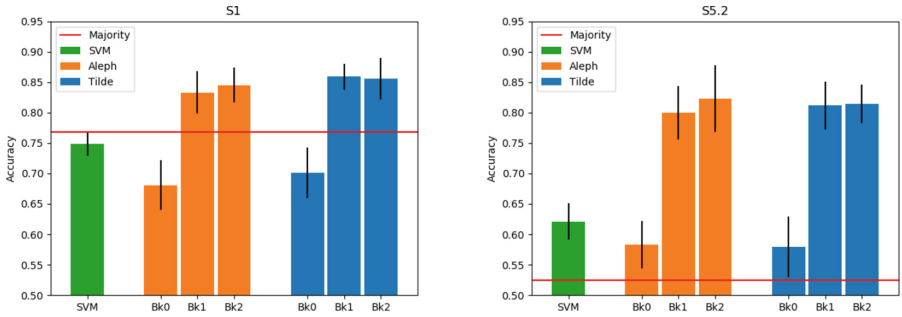

Fig. 3. Accuracy (mean and standard deviation) for all classifiers on S_1 and $S_{5.2}$

5.1 First Experiments

The performance of the classifiers is here evaluated using stratified 10-fold cross-validation. We see on Fig. 3 that:

- SVM together with Aleph and Tilde using BK_0 (refered to in the remainder as propositional learners) behave poorly, whatever dataset is used. The performance of those classifiers seem to depend also on the ratio P/N materialized by the performance of the majority classifier: they better model non-aggressive players. The propositional learners are nevertheless not able to reach a good performance. Some essential (relational) information is missing from the representation;
- Using relational BK (BK_1 and BK_2), ILP systems both significantly outperform the SVM system as well as ILP systems operating on raw propositional examples' representations (Sect. 3.2) on datasets shown, as well as others shown in the appendix. This demonstrates the relational essence of the problem. Their performances are around 82%, which can be considered as quite acceptable, considering E_4's consistency rate (88%), as shown in Sect. 3.1. Again, we evaluate here the relevance of the representation of examples and background knowledge more than the performance of propositional vs. relational learning systems.

– Aleph and Tilde have close performances whatever dataset and BK is used. It is in particular not possible to distinguish from an accuracy point of view models built using BK_1 or BK_2. Tilde may slightly outperform Aleph, although not on all sets.

5.2 Performance in Function of the Training Set Size

To evaluate the sensitivity of the learning systems to the training set size, we proceed as follows. For a given fold i, $1 \leq i \leq 10$, let us denote by $Test_i$ the test set and $Train_i$ the training set of the fold. For each fold i and proportion p, we sampled a stratified sample set $T_{i,p}$ of size $\frac{|Train_i| * p}{100}$ from $Train_i$. Each classifier learned for $T_{i,p}$ was evaluated on $Test_i$. Curves of Fig. 4 show the mean accuracy over 10 folds of the classifier learned with training set $T_{i,p}$ as a function of p for background knowledge BK_1 and BK_2.

As expected, the performance of all classifiers increases with the training set size, although we observe that ILP systems are able to reach a good performance with relatively few examples (only 10% of the training set). Hopefully, we do not observe any loss in performance for the largest training sets, which allows discarding overfitting.

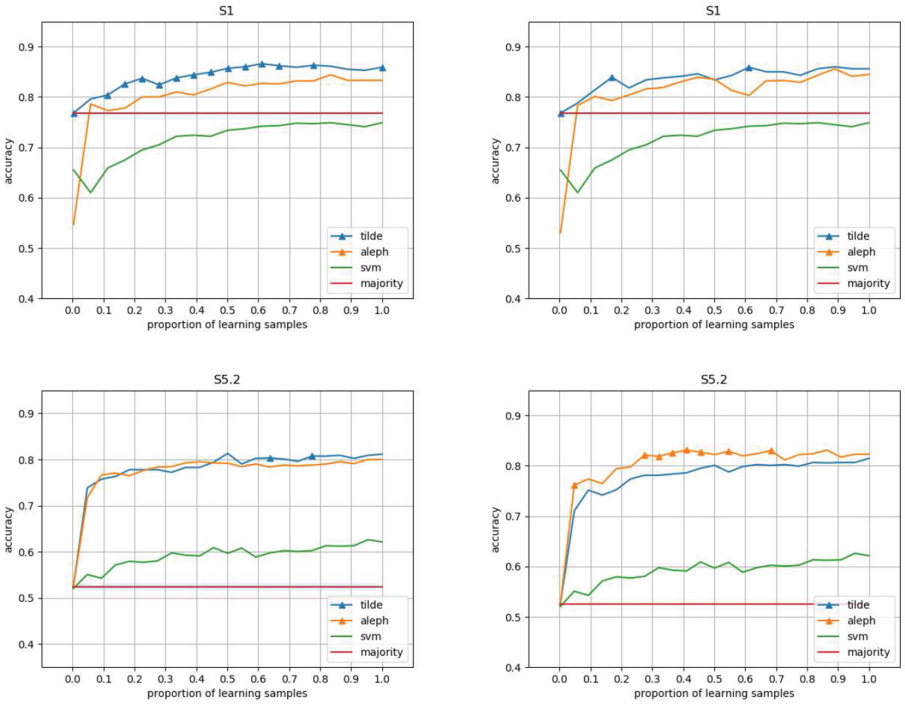

Fig. 4. Accuracy curves for BK_1 (left) and BK_2 (right) for datasets S_1 and $S_{5.2}$

We have again checked whether one ILP system significantly outperforms the other (paired t-test with 95% confidence). Significant differences are reported on the curve with a △. There are very few such differences in the performances of the two ILP systems, however Tilde is almost always the winner in such cases (see Fig. 10 in appendix), except for $S_{5.2}$ on BK_2. In this case, *list_honor*/3 allows to describe specific rules that Aleph can discover better than Tilde.

The difference w.r.t. the performance between BK_1 and BK_2 is tenuous whereas the differences in the theories and decision trees produced are obvious (see Fig. 5). These curves show the number of rules for Aleph and the number of nodes for Tilde for the same training set size as the previous curves.

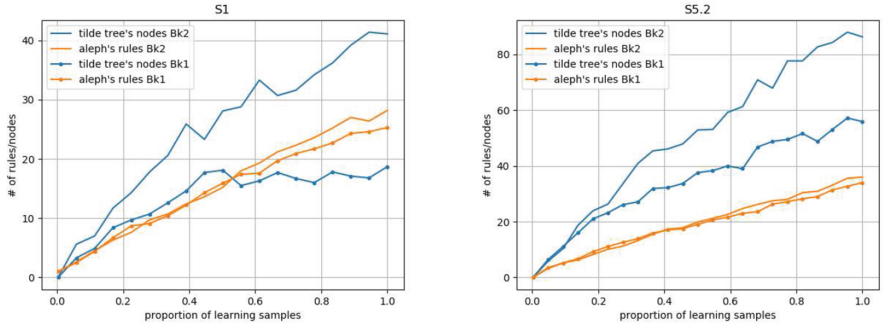

Fig. 5. Model complexity (number of nodes and rules) for S_1 and $S_{5.2}$ datasets and given different background knowledges (BK_1 and BK_2)

The only difference between BK_1 and BK_2 is the *list_honor*/3 predicate. It is used in about 80% of Aleph's rules and Tilde's nodes for BK_2. Aleph is less resistant to over-fitting (the complexity of the model increases regularly whereas the performance remains relatively stable after seeing about 70% of the training set). This is less true for Tilde, for which the model complexity stabilizes for S_1 and BK_1. Aleph's model complexity does not vary between BK_1 and BK_2 (about 25 rules for S_1 and between 30 and 40 rules for $S_{5.2}$). This is not the case for Tilde : the number of nodes of the learned DT is between 1.3 and 2 times larger for BK_2 than for BK_1.

6 Expert Validation

First of all, expert validations have been used to incrementally update the BK. In order to achieve this task, we made some experiments (not detailed in this paper) on intermediate BKs between BK_0 and BK_1 and we have presented the outputs of the models to the different experts. Their feedback (they validated or not the rules obtained) on the results help us to update at each step the current BK. For instance, seeing that the language used in rules was too poor led experts to advise

us to use other features such as the sum of the card numbers in ♡ and ♠. Experts also provided us with card combinations that make a hand more valuable (for instance King-Queen in a 5+suit) allowing to define and incrementally refine the predicate *plusvalue*. At the end of this process, we obtained BK_1 which is the first BK coupling good performances and positive feedback from experts regarding the learned rules. The experiments presented below are related to BK_1 and BK_2.

All experts noted that this experience had modified their strategy in this part of the game. The difference between the percentages of opened hands by E_4 on the same set (47.63 % the second time vs 55.72% the first time) confirms this point.

The following rule was unanimously validated. It reflects the fact that experts open with at least 6 cards in any suit.

```
r1: (Pos cover = 162 Neg cover = 1)
open(A) :- nb(A,B,C), gteq(C,6).
```

About the 12 rules produced by Aleph, experts gave the following opinion: 3 excellent rules (r1 and r2, r3 below).

```
r2 :  (Pos cover = 68 Neg cover = 3)
open(A) :- plusvalue(A), position(A,3).

r3 :  (Pos cover = 42 Neg cover = 4)
open(A) :- nb(A,spade,B), gteq(B,4), position(A,4).
```

r2 expresses that a player should open if he is in 3rd position and holds a good hand, whereas r3 means that a player should open if he is in 4th position and has more than 4 spades. When generating r3, Aleph discovered a famous Bridge rule known as 'the rule of 15': a player should open if he is in 4th position and if the number of HCP (11 points for all hands of this problem) plus the number of Spades is greater than or equal to 15. According to Bridge experts, the other rules were non-informative or even ridiculous for 2 of them.

The complexities of the models learned by Aleph are very similar among the experts (about 22 rules, see curves of Figs. 5 and 8 and Table 2 in the appendix). To get rid of the bias of the training set size, Table 2 in the appendix shows an evaluation of the learning model's complexity per expert depending on the background knowledge and the algorithm used for 4 training sets of the same size for each expert (703 training examples randomly selected from each set listed in the table). Set $S3$ that does not appear in the table obtained an average of 37.3 rules, a confirmation that this is a more difficult set (see Sect. 3.1). On the contrary, the complexity of the model learned by Tilde on sets labeled by expert E_1 (S_1) is significantly lower than those of every model related to the other experts.

The size of player's decision tree gives information about the player himself. For instance, E_1 has a scientific background and an analytical mind whereas E_4 has a more intuitive approach of the game. This is confirmed by the fact that

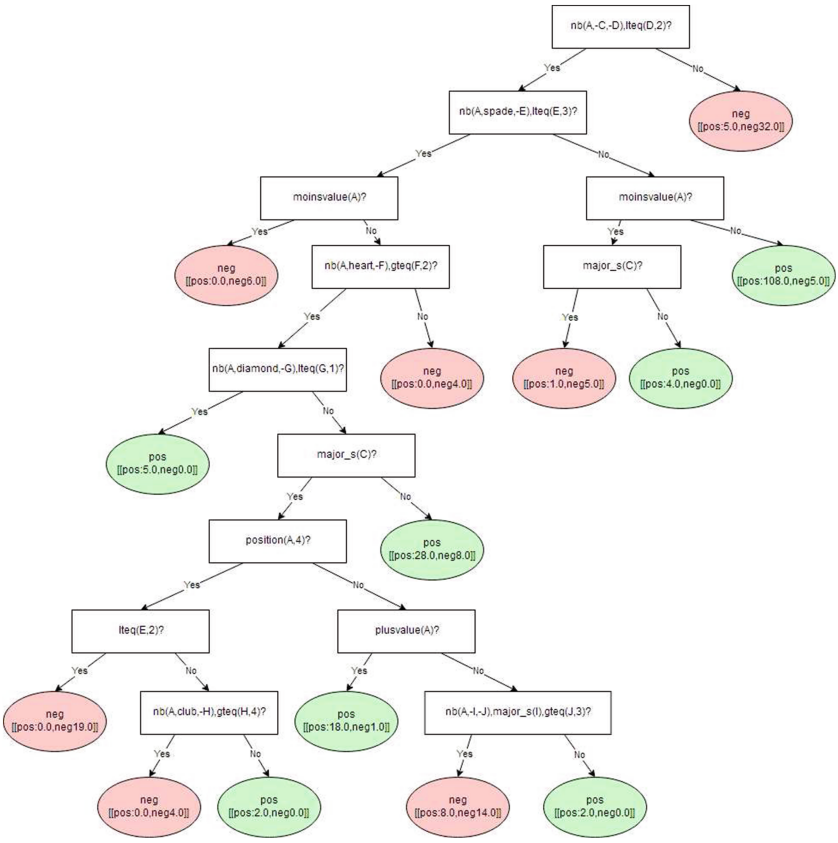

Fig. 6. Decision Tree on S_7 (279 learning samples) with BK_1

E_1's decision tree is two times smaller than E_4's decision tree ($S_{5.2}$ dataset). Moreover, the rules generated from E_4's set are of little interest to other players, being much too specific.

For this reason, we made a new experiment with expert E_1 on a new and smaller set S_7 (310 examples, 279 used for training), with a high level of difficulty since there is no hand in this dataset with more than 5 cards in a suit and the expert's position cannot be 3. The BK used is BK_1. We focus on E_1's validation on the decision tree produced by Tilde given Fig. 6.

Several rules associated with nodes have been described as 'excellent' by E_1. For instance, the rule associated with the root node translated as 'if the distribution of the hand is 4333, I pass' corresponds to one of the first criteria of his decision. The expert validated all the rules related to the nodes except one. Finally, the global vision of the tree appeared to him congruent with his approach of the problem. It is to be noted that before these experiments, the expert was not able to explain clearly his decision-making process on this type of opening.

In other words, Bridge experts make decision using a black-box approach, our methods allowing us to take a look inside the black-box.

7 Conclusions

Bridge has been subject to much research, mostly related to the conception of a Bridge robot. We refer to Paul Bethe's report [1] for extensive state of art on Computer Bridge. Like many programs playing incomplete information games, most current Bridge programs use Monte-Carlo methods which were initiated as early as 1949 by Nicholas Metropolis [13]. This method's adaptation to Bridge has been formalized in [7]. For more than 20 years, best Bridge robots have been competing at official World Computer-Bridge Championships (WCBC). The last 2 WCBC have been won in 2016 and 2017 by a version of AI Wbridge5 that has benefited from an enhancement of its Monte-Carlo simulations (see [17]). Some approaches using symbolic formalisms have been used in sub-problems of Bridge. For instance, in [6] the program Finesse that is coded in Prolog optimizes a card play sub-problem. In [9], authors use Horn clauses to model a bidding rule base. More recently neural approaches (e.g. [8,19]) have been successfully used to automate a limited version of the bidding process. However, these approaches do not provide explanations and do not allow to deviate from the rules as a human does for instance in our opening problem.

The methods displayed in this article for the opening bid problem are currently being extended to other Bridge situations, some of them requiring an upgrade to multi-class learning. Future work related to a Probabilistic Inductive Logic Programming (PILP) framework [4] seems very appropriate since probabilistic reasoning is central in Bridge. Another important extension of the current work is the ability to invent new predicates, using various techniques such as relational pattern mining [5], or more recent approaches in a Statistical Relational Learning context [10,11]. As stated in [18], Bridge is a great challenge for AI and much work related to the definition of a Bridge AI remains to be done. The design of a hybrid architecture including recent numerical, probabilistic, symbolic machine learning modules is currently under process in a project called νBridge (new version of the AlphaBridge project), this work is the prime stone of the ILP-PILP module that will be central in νBridge. In such a framework, ILP systems could be used as a front-end to statistical learning systems in order to generate appropriate representations of examples that take into account some relational traits of the learning problem at hand.

References

1. Bethe, P.: The state of automated bridge play. PDF (2010)
2. Blockeel, H., et al.: The ace data mining system user's manual. https://dtai.cs.kuleuven.be/ACE/doc/ACEuser-1.2.16.pdf
3. Blockeel, H., De Raedt, L., Jacobs, N., Demoen, B.: Scaling up inductive logic programming by learning from interpretations. Data Min. Knowl. Discov. **3**(1), 59–93 (1999)

4. De Raedt, L., Kersting, K.: Probabilistic inductive logic programming. In: De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S. (eds.) Probabilistic Inductive Logic Programming. LNCS (LNAI), vol. 4911, pp. 1–27. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78652-8_1
5. Dehaspe, L., Toivonen, H.: Discovery of frequent datalog patterns. Data Mining Knowl. Discov. **3**(1), 7–36 (1999)
6. Frank, I., Basin, D., Bundy, A.: Combining knowledge and search to solve single-suit bridge. In: Proceedings of AAAI/IAAI, pp. 195–200 (2000)
7. Ginsberg, M.L.: GIB: imperfect information in a computationally challenging game. J. Artif. Intell. Res. **14**, 303–358 (2001)
8. Ho, C.-Y., Lin, H.-T.: Contract bridge bidding by learning. In: Proceedings of Workshop on Computer Poker and Imperfect Information at AAAI Conference on Artificial Intelligence (2015)
9. Jamroga, W.: Modelling artificial intelligence on a case of bridge card play bidding. In: Proceedings of the 8th International Workshop on Intelligent Information Systems, pp. 267–277 (1999)
10. Kazemi, S.M., Poole, D.: RelNN: a deep neural model for relational learning. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (2018)
11. Kok, S., Domingos, P.M.: Statistical predicate invention. In: Proceedings of the Twenty-Fourth International Conference on Machine Learning, ICML 2007, pp. 433–440 (2007)
12. Mahmood, Z., Grant, A., Sharif, O.: Bridge for Beginners: A Complete Course. Pavilion Books, London (2014)
13. Metropolis, N., Ulam, S.: The Monte Carlo method. J. Am. Stat. Assoc. **44**(247), 335–341 (1949)
14. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
15. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
16. Srinivasan, A.: The aleph manual (1999). http://www.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/
17. Ventos, V., Costel, Y., Teytaud, O., Ventos, S.T.: Boosting a bridge artificial intelligence. In: Proceedings of International Conference on Tools with Artificial Intelligence (ICTAI), pp. 1280–1287. IEEE (2017)
18. Ventos, V., Teytaud, O.: Le bridge, nouveau défi de l'intelligence artificielle? Revue d'Intelligence Artificielle **31**(3), 249–279 (2017)
19. Yeh, C.-K., Lin, H.-T.: Automatic bridge bidding using deep reinforcement learning. In: Proceedings of the 22nd ECAI, pp. 1362–1369 (2016)