A Markovian model for the Game of Truco

Lana Rossato
Department of Applied Computing
Universidade Federal de Santa Maria
Santa Maria, Brazil
lbrossato@inf.ufsm.br

Luis A. L. Silva
Department of Applied Computing
Universidade Federal de Santa Maria
Santa Maria, Brazil
luisalvaro@inf.ufsm.br

Joaquim Assunção
Department of Applied Computing
Universidade Federal de Santa Maria
Santa Maria, Brazil
joaquim@inf.ufsm.br

***Abstract*—In recent years, the advancement of Artificial Intelligence (AI) for games is notorious. However, despite achieving the expected results, many AI techniques require considerable computational power, which is not viable when games are running in more basic computing platforms. Observing the need of creating and using light-weight techniques, this work uses a Markovian structure for the development of an agent to play the card game of Truco. The proposed modeling represents the natural order of the card strengths in the game. It is also separated into modules according to the game rules and optimized according to the game state-space. The model also learns at each game action, reinforcing decisions, and improving future responses.**

***Index Terms*—Markovian Models, Statistical Learning, Game of Truco**

I. Introduction

Games are an interesting environment for artificial intelligence (AI), In addition to being widely used for entertainment, games can also represent real-world problems, where solutions to these synthetic problems can be explored to improve solutions for the real-world ones [1]. In particular, card games provide an interesting environment for research in AI, since there are several adversities, such as partial view of the game, due to the opponent's cards that are not visible, and random distribution, when the cards are shuffled at the beginning of the game.

Card games are different from deterministic games, which have a total view of the game, such as Chess, Go, or Checkers. The card game scenario has become a challenge [2], since human players still have the advantage over state-of-the-art artificial agents. A few works have made significant advances in partial-view games, such as Texas hold'em Poker [3], [4] and StarCraft II [5], causing the performance gap between humans and artificial agents to narrow. However, state-of-the-art agents are based on deep learning models, which require an enormous amount of computational power, making it unfeasible for ordinary computers.

In this context, a light-weight Markovian-based technique is used to propose a platform model to the Game of Truco. In other words, a model that can be used to play the game effectively and efficiently, in terms of computational requirements, yet simple and open to be improved when integrated to other AI techniques. The Markovian structure is probabilistic and its decisions are made only on the cards available (player and opponent), and the mode of the game (*Truco* or *Envido*).

II. Background

A. The game of Truco

Among the variations present in Brazil, the game rules used in this work are those of "Truco Gaudério", most commonly played in the southern regions of South America. Each player receives three cards and the objective is to earn more points than the opponent. The first player (or team) to achieve the defined limit of points (usually 24) is the winner. Each round consists of each player showing a card and the winner is who has the strongest card. Each hand, set of one to three rounds, is worth one point, yet that value can change with the bet of *Truco*. After the bet, the value of the hand can be increased, as shown in Table I, and these points are directly reflected on the final score. This increase is achieved gradually, with bet acceptance (similar to call in Poker) and bet raised, called "*Retruco*" and "*Vale 4*".

TABLE I. Bets and raises.

Bet	1st^a Raise	2nd^a Raise
Envido	Real Envido	Falta Envido
Envido	Falta Envido	-
Real Envido	Falta Envido	-
Falta Envido	-	-
Flor	Contra Flor	Contra Flor e Resto
Truco	Retruco	Vale 4

Envido and *Flor* are disputes parallel to the game. They both add points to the final score and both occur at the beginning of the hand, before playing the first card. In *Envido*, a number is calculated from the cards of each player and whoever has the highest number, wins. The total *Envido* points is just the sum of two cards plus 20 points (if the same set)[1]. As well as the *Truco* bet, the *Envido* can be raised to +2 points (the other player calls Envido), +3 points (*Real Envido*), and +W points, being W the number required for the winning player to win the game. The same logic is applied to *Flor*, except it cancel the *Envido* bet. Table I shows the available bet modes. Table II shows the strengths for all the cards in each bet mode.

Just like Poker, it is possible to bluff, so if an opponent propose a bet and the player do not accept it, the opponent might win with weak set of cards. Such characteristic makes difficult to create a rule-based AI since a human player can

[1]Except the cards 10, 11, and 12

TABLE II. Cards' strength.

Strength	Truco	Envido/flor
1º	$1\spadesuit$	Any two cards of the same set (20 points)
2º	$1\clubsuit$	$7*$
3º	$7\spadesuit$	$6*$
4º	$7\diamondsuit$	$5*$
5º	$3*$	$4*$
6º	$2*$	$3*$
7º	$1\diamondsuit, 1\heartsuit$	$2*$
8º	$12*$	$1*$
9º	$11*$	$10*, 11*, 12*$ (0 points)
10º	$10*$	
11º	$7\heartsuit, 7\clubsuit$	
12º	$6*$	
13º	$5*$	
14º	$4*$	

easily exploit an AI that always play according to the hands' strength. Further information regarding the Truco game can be found in [6].

B. Markov models

A Markovian model is a special type of stochastic process that can be applied to almost any system [7]. Such models can represent all the states of a system, with assigned transition probabilities among the states. The system can only occupy one state at time and the probability of being in a current state only depends on the previous state. Formally, we can say that the process can be represented by a stochastic variable in time $\{x^{(t)}, t \in T\}$, being $x^{(t)}$ the occupied state in a given time t.

The states of a Discrete Time Markov Chain, as used in this work, are represented by a set of N variables T times, defining the total number of states N, and the time set T. The set of probabilities that takes a state x_n to all other states in time t has a total sum corresponding to 1. Fig. 1 shows an example of a Markov Chain for the weather states' *Sunny*, *Cloudy*, and *Rainy*. The probability of passing from Cloudy to Sunny if 40%, to Rainy is 50%; and to remain cloudy is 10%. If such model represents the probability for a given day, we can achieve the probability directly from the transition probability. For two days ahead we need to calculate using the Chapman Kolmogorov equation. Fig. 1 shows an example of Markov Chain for a three-state weather forecast.

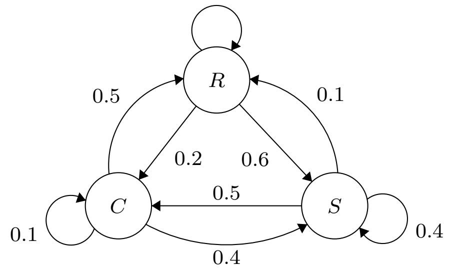

Fig. 1. Example, Markov Chain for a three-state weather forecast.

The same Markov can be represented as a transition probability matrix. By default, a transition from X_1 to X_2 is represented as line to column.

TABLE III. Table representation for Figure 1.

	C	R	S
C	0.1	0.5	0.4
R	0.2	0.2	0.6
S	0.5	0.1	0.4

Thus, if the weather is Cloudy, the probability to be rainy in the next time (t) is 50%. two steps ahead ($t + 2$) is 19%. After a few steps, the probability becomes steady at 24%. These probabilities are useful to choose an appropriate action for turns ahead of the current state. I.e., as the *Truco* game has 3 turns in each hand, it is necessary to consider prior probabilities for the 2nd and 3rd turns.

III. Related Works

Ambekar et al., [3] developed a system to help players to intelligently bet. The system gives the player the probability of winning based on the available cards, the only visible features available in the decision-making state of the game. Relying on a Bayesian classifier, training and test data were used to predict the class in which the hand cards fit. To identify the players' profile based on the data regarding their game moves, [8] assumed that the players' characteristics influence their performance leading to the identification of their profile. To achieve this goal, the authors used the multiple linear regression techniques to analyze the data and to find the characteristics of the players. Afterwards, they used the k-means algorithm were used to form groups of players along with their playing profiles. Finally, the authors studied the correlation between game-playing characteristics and selected the variables for future analysis. Thus, the multiple linear regression model was explored, revealing that there are significant effects on variables that were considered dependent, such as specific game missions. With clustering, it was possible to identify three types of players: beginner, intermediate and advanced, being players' groups reflecting distinct behaviors and actions in the game.

To develop an agent for card games, [9] used a combination of Case-Based Reasoning (CBR) and clustering techniques applied to the reuse of game actions present in cases. Supported by the K-means algorithm, the reuse criteria, for choosing game actions, were executed by the agents considering the group of cases present in the case base. In particular, these groups were related to the different game playing states in which game actions were taken according to the analysis of past game situations recorded as cases. The reuse of the past game actions were extracted according to different criteria: 1, majority rules. 2, choice of cluster/action by probability (hand by hand). 3, choice of action/cluster with a higher probability of victory, and 4, choice of cluster/action leading to the highest gain of points. All these criteria were applied to CRB only and CBR+clustering bots. As a result, it was found that the "highest gain of points" with clusters was the best combination, followed by "probability of victory", which is partially part of the first. Using Hidden Markov Models, [10] proposed a method for recognizing facial expressions in interactive and cooperative games. To this end, the modeling took into account the parts of the players' face to be analyzed, where this built-in modeling aimed to provide better estimates of players' profiles. To the training step, several images were used where the tested algorithm presented promising results. Analyzing human players who were already in the training set, the algorithms reached an accuracy of 92%, while in new players' faces, achieved 84%. Using the same tool, [11] studied the recognition of gestures that served as input to the game. The model was fast in the recognition of the players' gestures, but it was not enough for actual real time responses. The recognition accuracy depended directly on the time spent on training and the number of states in the model.

Our model incorporates a few characteristics of these works, similar to [3], probability techniques are explored, which might help players improve their bets. The model is designed to learn at each hand, which might be improved to learn players' profiles and respond accordingly (similar to [8]). Furthermore, the model is based on a probability directly linked to the force of the cards, which is a perfect mapping of the "probability of victory" described in [9]. Finally, the overall model is Markov based, an approach similar to [10], [11]; yet lightweight since we are not doing any visual computation.

IV. Model

Our model is divided into four components, one for each main mode of the game (*Truco*, and *Envido*), one specifically to the *Flor* mode, and one to control the order of the played cards according to its strength (*play card*).

- Truco. Controls the decision to call, accept or deny a bet. In addition, depending on the Markov probability, it can lead to an increase or a withdrawal.
- Play card. The order of the cards to be played in each round. For example, a player with $4*, 4*, 1\spadesuit$ should never play two $4s$ in a row.
- Envido. The decision to start a bet, accept, deny, raise or fold.
- Flor. Similar to Φ, except that the *Flor*'s points are based on three cards.

Since the global model is modular, any changes or improvements can be made locally. Errors in one component will not affect the others. Furthermore, being the global state a combination of 4 local states, the memory usage is improved. Each time an action happens on the game, the result is reflected on the respective component. I.e., if the Bot loses, the Markov transition probability decreases, otherwise it increases.

The size of each Markov is based on the total state-space for the game. Since all the states are possible and each Markov is previously filled, there is no sparsity. The next Subsection shows how it was calculated.

A. State space

Truco mode

Considering the total number of cards to play Truco (40), the total number of hands is:

$$C(40,3) = \frac{40!}{(40-3)! * 3!} = 9880 \qquad (1)$$

As seen in Section II-A, however, few cards have the same strength regarding the suit, they are all 2..6 and 10..12. For instance, a $5\heartsuit$ has the same strength of a $5\clubsuit$, $5\spadesuit$,or $5\diamondsuit$; hence it does not matter the number, they make the same game state. From 40 cards, we have in fact 14 levels (see Table II). The combination of 14 levels, with repetition, is given by:

$$C(14,3) = \frac{(14+3-1)!}{(14-1)! * 3!} = 560 \qquad (2)$$

However, not all cards on the same level allows repetition, as the cards 1 and 7 have 3 different levels. For instance, $1\spadesuit > 1\clubsuit > (1\heartsuit = 1\diamondsuit)$. Thus, for instance, $1\spadesuit$ is a unique card that should not be counted. All the combinations counting the unique cards (a card in its own level) and the semi-unique (Less than four cards for a level, e.g., $1\heartsuit$ and $1\diamondsuit$) makes 58 states. Which bring a total of 502 states ($560 - 58 = 502$). The final model is represented by Fig. 2, where:

- H_n represents a local state. A unique hand of cards for the *Truco* mode.
- μ is the average weight for any game, i.e., 50% chance to accept a call.
- σ and θ are vectors with weights to the correspondent hand.
- $1 - \sigma$ and $1 - \theta$ are the complementary values for the vectors σ and θ.
- a and $1 - a$ are the weight (transition probability) from the strongest hand to the weakest and vice-versa.

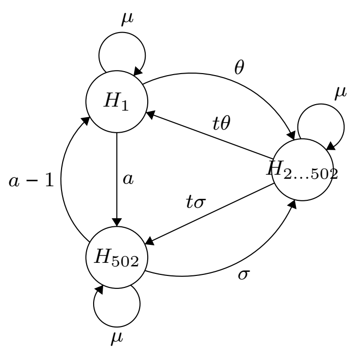

Fig. 2. Adaptive Markov chains for Truco.

It is possible to observe that the resulting structure is not a real Markov. Instead, it is an adaptation of the formal chain. However, the single state to state (1 to 1) only happens when an opponent plays the last card without any bet. Which triggers a deterministic action, accept, or bet if the hand is better. But also keeps the stochastic characteristic with respect to bluffs (if the hand is worst, what are the chances for a gamble?).

Each time a set of cards is visible a set of states is separated from the chain. For instance, if an opponent plays the 1♣, the probability for any decision will be selected based on all possible combinations with such a card. Thus, cutting all the states without 1♣. In the end, the main Markov properties remain. All output probabilities sum 1 and the next state (action against an opponent hand) only depends on the current state (player hand).

Envido mode

The *Envido* and *Flor* modes follow the same logic previously described. The only difference is the triggers for the game condition (e.g., a player cannot bet *Envido* after the first card) and the state-space. Regarding *Envido*, there are 29 states; $C(8,2) = \frac{8!}{(8-2)!*2!} = 28$ plus any lower combination without two of a suit (¡20 points, all considered bluff and treated as one state). Regarding *Flor*, there are 64 states; $C(8,3) = \frac{8!}{(8-3)!*3!} = 56$ plus 8 combinations with cards that do not value any *Envido* point (10,11, and 12).

Play card

The *Play Card* component is the only non-Markovian component. This component is stored directly in 2 matrices. The first, being responsible to store the probability of a card be the *the lowest*, *the average*, or *the highest* card in a hand. For instance, 1♠ will always be the highest card in a hand, 6♡ almost always will be the lowest card, and 2♡ will almost always be an average card. The playing order is important, for instance, if a player has two $5s$ and one 1♣, usually, it is a bad choice to play 1♣ last. Furthermore, the order in which the opponent plays his/hers cards is important for a player to react correctly. Thus, another matrix stores the combination of orders among players.

V. Experiments and Conclusion

Currently, each time a bet is won, there is a 10% increase on the probability regarding that situation (i.e., same cards, same bet). Otherwise, the probability decreases by 10%. Fig. 3 shows the change in the transition probability matrix values. Each pixel is a specific combination of cards for the *Truco* mode. The highest probabilities are represented in green, the lowest in brown. The strongest game is in the last row, the weakest in the first. A similar representation is used for the *Envido*/*Flor* mode.

It is possible to see changes in the heatmap B, each pixel represents one or more cases of success or failure regarding a bet. Thus, the probabilities are constantly changing according to the opponent. However, due to the number of states, the change is too slow to improve the bot in real-time. Hence it is necessary a function that spreads the probability around the current state. In other words, if a set of cards X loses for a set of cards Y, every similar game should be updated with gradual probability change. Nonetheless, even without a proper function to propagate the changes, the model so far achieved 28% of victories against the Truco player agent described in [9] (56/200 matches, against 4 bots). These are satisfactory results considering the number of interactions in a Game, and the current state of the model, with no learning propagation

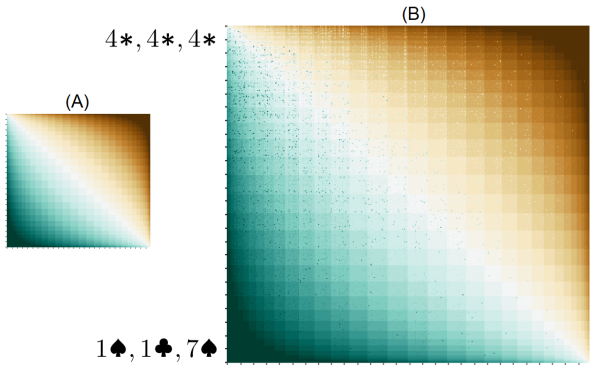

Fig. 3. *Truco* mode. Heatmaps showing the initial probability distribution (A) and the distribution after 100 matches (B).

or even previous training. Furthermore, the computational power required is minimal ($O(n)$), where n is the number of reachable states. This allow us to port the solution to any device. As future work, we will use different propagation methods in order and train the bot to calibrate the learning rate (instead of a fixed 10%). Finally, the bot will be tested against human players.

References

[1] S. Balan and J. Otto, *Business Intelligence in Healthcare with IBM Watson Analytics*. North Charleston, SC, USA: CreateSpace Independent Publishing Platform, 1st ed., 2017.

[2] J. Niklaus, M. Alberti, V. Pondenkandath, R. Ingold, and M. Liwicki, “Survey of artificial intelligence for card games and its application to the swiss game jass,” in *2019 6th Swiss Conference on Data Science (SDS)*, pp. 25–30, 2019.

[3] G. Ambekar, T. Chikane, S. Sheth, A. Sable, and K. Ghag, “Anticipation of winning probability in poker using data mining,” in *2015 International Conference on Computer, Communication and Control (IC4)*, pp. 1–6, 2015.

[4] I. Watson, S. Lee, J. Rubin, and S. Wender, “Improving a case-based texas hold’em poker bot,” in *2008 IEEE Symposium On Computational Intelligence and Games*, pp. 350–356, 2008.

[5] G. Synnaeve and P. Bessière, “Multiscale bayesian modeling for rts games: An application to starcraft ai,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 8, no. 4, pp. 338–350, 2016.

[6] L. L. Winne, *Truco*. Ciudad Autónoma de Buenos Aires Ediciones Godot, 2017.

[7] W. J. Stewart, *Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling*. Princeton University Press, 2009.

[8] S. Benmakrelouf, N. Mezghani, and N. Kara, “Towards the identification of players’ profiles using game’s data analysis based on regression model and clustering,” in *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 1403–1410, 2015.

[9] G. B. Paulus, J. V. Carvalho Assuncao, and L. A. L. Silva, “Cases and clusters in reuse policies for decision-making in card games,” in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 1361–1365, 2019.

[10] Xiaoxu Zhou, Xiangsheng Huang, and Yangsheng Wang, “Real-time facial expression recognition in the interactive game based on embedded hidden markov model,” in *Proceedings. International Conference on Computer Graphics, Imaging and Visualization, 2004. CGIV 2004.*, pp. 144–148, 2004.

[11] L. Kratz, M. Smith, and F. J. Lee, “Wiizards: 3d gesture recognition for game play input,” in *Proceedings of the 2007 Conference on Future Play*, Future Play ’07, (New York, NY, USA), p. 209–212, Association for Computing Machinery, 2007.