Investigating Case Learning Techniques for Agents to Play the Card Game of Truco

Ruan C. B. Moral
Undergraduate Program in Computer Engineering
Federal University of Santa Maria – UFSM
Santa Maria – RS, Brazil
ruancmoral@gmail.com

Gustavo B. Paulus
Graduate Program in Computer Science
Federal University of Santa Maria – UFSM
Santa Maria – RS, Brazil
gustavobpaulus@gmail.com

Joaquim V. C. Assunção
Applied Computing Department
Federal University of Santa Maria – UFSM
Santa Maria – RS, Brazil
joaquim@inf.ufsm.br

Luis A. L. Silva
Graduate Program in Computer Science
Federal University of Santa Maria – UFSM
Santa Maria – RS, Brazil
luisalvaro@inf.ufsm.br

***Abstract* - Truco is a popular game in many regions of South America; however, unlike worldwide games, Truco still requires a competitive Artificial Intelligence. Due to the limited availability of Truco data and the stochastic and imperfect information characteristics of the game, creating competitive models for a card game like Truco is a challenging task. To approach this problem, this work investigates the generation of concrete Truco problem-solving experiences through alternative techniques of automatic case generation and active learning, aiming to learn with the retention of cases in case bases. From this, these case bases guide the playing actions of the implemented Truco bots permitting to assess the capabilities of each bot, all implemented with Case-Based Reasoning (CBR) techniques.**

***Keywords* - Case learning, Case-based reasoning, Card games, Truco game.**

I. Introduction

Computer games are officially a sport (publicly known as E-Sports). These games are capable of moving thousands of people to attend annual competitions, such as the League of Legends World Championship in 2013 [1], for example. In this context, the effort to design techniques aimed at supporting the development of more fun and competitive computer games has allowed achieving significant advances in Artificial Intelligence (AI). As such games evolve, players also expect game situations that are closer to reality, with interactions between agent players that resemble the behavior of real people [2].

For agents inserted into games to act similarly to humans, it is relevant to investigate ways of learning what the agent should perform in each problem situation of the game. In this line of research, Case-Based Reasoning (CBR) [3] is an important AI technique that reflects human decision-making behavior in the resolution of many complex problems. In different works in the area of computer game development [4], a case-based agent (described in this article by the term BOT) recalls a past game situation (a past case) similar to a current game situation (a current case problem to be solved). Once a concrete problem-solving experience is retrieved from memory, materialized as a case base in CBR, the agent reuses the decision made in the past to solve the current problem. For a case-based agent to perform actions in a competitive game, that agent can initially use demonstrations of game actions performed by human players, where demonstrations of various kinds can be recorded as cases in the case base [5]. In a complex game environment, however, it is exhausting for game developers to demonstrate every possible problem and solution that can be required by an agent or even to program a game script reflecting a general problem-solving behavior. As explored in this article, it is necessary to have techniques that enable the BOT to learn how to act even when limited initial knowledge about the problem-solving is available. In many senses, new cases have to be retained in case bases to support the decision-making process of agents in various kinds of computer games.

To explore case learning techniques for games, a CBR system applied to the Truco's card game was developed. Truco [6] is a widely practiced card game in the southern regions of South America, although some of its rules may vary according to the region, or country, in which it is played. In many ways, this game has characteristics that are similar to the game of Poker and other card games. Unlike Poker, however, research involving the game of Truco in the literature is still limited [7, 8]. In the field of research regarding CBR learning for competitive games disputed between pairs of opponents, the Truco game presents a fun and motivating environment where a set of research challenges for AI can be addressed, such as how to approach continuous learning, for example.

In this project, a set of 513 hands of Truco matches played between two human players were initially collected and stored in a case base. From these cases, a BOT observed how human players acted in the past to then reproduced these actions in the current games. However, the main problem with this BOT is that 513 records of Truco hands may be a limited number of gaming experiences compared to other game applications similar to Truco. As described in [9], for instance, 50,000 Poker hands were used to guide agents in making decisions in that game. Using the set of 513 cases of Truco games, therefore, this work investigates alternative case retention approaches to support intelligent agents to play the game of Truco. Among other goals, this initial case-base made it possible to investigate different forms of case retention: a) the retention of new cases and b) the retention of new cases with substitution, in addition to c) the use of random game actions to generate new problem-solving experiences to be retained. Such techniques were performed whenever the agent detected that it did not know how to solve a current problem in the game. As this BOT may also need to perform game actions as similar as possible to actions that human players would perform, a d) tutoring strategy based on the active learning technique [10, 11] was also explored in the case retention investigation presented in this work. In this case, when experiences capable of solving the current problem were not found in the case-base used by the agent, and only in such situations, this agent requested the help of a human player (the first author of this paper), who indicated what this BOT should do. In this way, a new and concrete problem-solving experience was generated and inserted into the case-base, permitting the agent to develop new Truco moves.

To assess the effectiveness of the different case learning techniques investigated in this work, a set of Truco matches between agents was performed. In doing so, agents' game decisions were based on a) the case bases automatically generated and based on b) the initial case base (the BASELINE case base) collected in this project. In addition, a competition between agents playing with case bases generated from each one of these retention techniques was carried out to determine which agent have used the most competent case base.

II. Background to this Work

The development of AI techniques for games has lately shown great effectiveness mainly with respect to deterministic board games like Chess and GO. Such games involve the dispute between two players, where the agents have perfect information about the game. To have perfect information means that at each moment of the game an agent can observe the environment and know the opponent's moves. In general, these games may not be considered as complex as Poker, for example, which is a game with imperfect information. This is because there is information on games of Poker that is only observed with the development of the game, in addition to the luck factors involved [12]. For example, each player in the game of Poker receives shuffled cards. In addition, each agent only has information from their cards, usually not knowing the opponents' cards. When Poker games are mostly played with entertainment purposes, bluff techniques are also often explored, where the quality of the received cards is not enough to make various good decisions.

A. *CBR in digital games*

Case-Based Reasoning (CBR) [3] is focused on the development of intelligent agents in a wide variety of applications, as well as being used to support the development of agents immersed in computer games. The basic idea of CBR is to recall a previous situation which resembles a current one and then reuse past case-based knowledge to find a solution to the current problem. In CBR, a central idea is that similar problems have similar solutions, and that these problems tend to repeat themselves over time. The CBR cycle is commonly described as a four-step process (4R cycle). The retrieval step uses the characteristics of a current problem, where a query on a case-base is performed. This allows retrieving the past cases that most resemble the current problem. The reuse step uses the most similar cases retrieved for a given query. This step involves reusing the past case solutions to solve the new problem. The revision step involves checking the quality of the reused solution, to avoid proposing solutions that are ineffective or even impossible to be applied to. Finally, the retention step involves analyzing whether the current case, solved from the reuse and revision of the past solutions, is relevant to be remembered to support the solution of new future problems.

The retrieval of cases from the case-base is one of the main steps of the CBR cycle. A description of the current problem is created where the characteristics, considered relevant to find a solution to this new problem, are identified in the query formation. With this query description, a search in the case-base is carried out to identify which past problem-solving situations are the most similar to the current problem to be solved. The case retrieval strategy most commonly used in CBR systems is based on the K-Nearest Neighbor algorithm. There are several functions for the assessment of the case similarity between current and past case situations. These functions can be organized according to similarities that are local, computing similarities between individual case attributes, and similarities that are global, where an amalgamation function computes similarities in the level of cases. In CBR, a common approach is to apply different local similarity functions to different kinds of attributes, and then to use the Euclidean distance function to measure similarities between query and past cases, as explored in this work. Once such similarity is computed, the retrieved cases are organized in a ranking order based on the global similarity results.

At the end of the 4R CBR cycle, it is necessary to analyze if the solved current problem should be remembered as to support the resolution of new problems in the CBR system. In this case, case retention in the case-base involves a process of learning, such as learning how to solve a new problem that does not yet exist in the case base. The objective of retaining case-based knowledge about how to approach new problems is to constantly update and improve the competence of the case-base [13, 14], permitting the CBR system to approach a larger number of problem situations.

As described in [4], research involving the exploration of CBR techniques in games can be concentrated in categories: "classic board games", "adventure games", "team sports", "real-time individual games," "real-time god / management games", and "strategy-based games in shifts" (discrete / turn-based strategy). In quite challenging scenarios, CBR techniques have recently been explored to solve problems in the category of RTS games. For example, [15] describe case acquisition strategies to support the development of agents aimed at RTS games. In addition, [16] explore CBR in the automated modeling of learning processes and adaptation of opponents in the RTS game called GLest. In another work, [17] explore CBR to perform a more effective army composition in the RTS game StarCraft. In the card game of Truco, this article also explores learning techniques like the ones cited here.

In more classic game environments, CBR is used in [18] to model and detect a player's skill level in the Tetris game. The objective is to dynamically adjust the difficulty level of the game according to the skill of the player. Based on this adjustment, the approach involves improving player satisfaction to maintain his/her engagement with the game. [19] present the CHEBR system containing an agent capable of learning how to play checkers. CHEBR uses the "Automatic Case Elicitation" (ACE) [20] approach to building a case base. In this learning technique, similar to what this article explores, a query on the case base is performed. If a past case to solve the current problem is not found, a random attempt to solve the problem is performed. Then, the quality of this trial is assessed. From it, new relevant cases are determined and stored in the case base. [9] use CBR not to play, but to present a more challenging game for the players. COMETS is an agent capable of watching user's moves in the Space Invader game. From this observation, the agent can detect plans or patterns that may appear, allowing the identification of the future actions of these players. The idea is to anticipate what a player can play to generate a more challenging game.

Designed to play Texas Hold'em, the most common style of Poker, [21] describes the CASPER system (CASe-based Poker playER). This system explores CBR techniques to make decisions on the types of strategies that must be applied during each stage of a Poker match. These strategies consist of actions that must be performed given the stage of the game: raise, bet, fold, take no action or accept the bet, among others. In addition, there is also a classification of these plays between honest or tricky ones. Thus, the developed agent can play with different styles, which have been divided into aggressive or passive. Like CASPER, SARTRE [22, 23] was designed to use CBR in the game of Poker to make betting decisions based on past cases. Unlike CASPER, which was developed for the 10-player Texas Hold'em style, SARTRE is geared towards matches between two players, just as explored in this work.

In particular to the Truco game context, [8] explore image processing techniques to analyze the visual messages often explored in multi-player Truco matches. Based on alternative CBR reuse criteria described in the literature [22-24], a two-step CBR reuse model was proposed in [7]. There, the integrated use of CBR and clustering techniques was explored to assess Truco cases stored in the case bases in terms of their decision-making game states along with the game actions advanced in the recorded Truco scenarios. The paper also investigates the exploration of different reuse criteria (probability lottery, majority rule, best-outcome) in the choice of which case groups and game actions to reuse as to support agents to solve Truco problems in new game situations. This paper considers this research and expands it in direction to the investigation of alternative case learning approaches.

B. CBR learning

Intelligent agents can be in constant learning. In the case of CBR systems, the knowledge of the system is stored mainly in the case base, among other repositories of knowledge [25]. Exploring such case bases, the CBR system can remember past cases to reuse successful solutions to current problems. When a problem is solved via CBR, for example, this problem-solving experience can be recorded in the case base, possibly improving the competence of the case base. In general, the CBR system should have learning resources to save new cases in the case base, but it should also have resources that allow forgetting failure situations, redundant or unnecessary, automatically removing such cases from the case base.

CBR systems are traditionally built from previously organized case bases. In situations where such a case base is not available or is available although it is partially complete, it is possible to explore the ACE strategy [20]. In many problems, using this strategy allows a system to develop its knowledge automatically. This is done through trial and error, where the system has no prior knowledge. The ACE flow consists of carrying out the action proposed by a set of retrieved cases according to the maximum number of cases and a certain minimum similarity threshold. If it is not possible to retrieve any past situation (or case) according to the defined criteria, random game actions can be performed. As this random action can be ineffective, the result obtained with this game action must be observed. At the end of each game interaction, some metrics are used to verify the effectiveness of each performed game, where the result of each action employed is evaluated, generating "rewards". In general, this allows one to assess what the best cases are. Thus, it is possible to select only the best solutions and perform the removal of actions with poor performance as in [26]. Due to the ability to explore such game attempts, this technique is suitable for learning in domains where the outcome of the games is easily observable. As random game actions are made, it is possible to perform different experiments and then select the best ones.

In the scope of learning for games, agents should be able to interact in the game as realistically as possible. In many ways, these agents could act like a human player would act given a problem situation. In this scenario, a technique commonly used for the development of agents in CBR systems is the "Learning by Observation". Also known as "Imitation Learning" or "Demonstration Learning" as described in [11], observation learning is a very common human learning strategy, where an example (used as a model) is sought for information on how to perform a decision for a problem situation. For instance, an agent learns to move a bicycle pedal in a certain way by watching a teacher to demonstrate the movements or sequence of steps that should be performed. In demonstration learning, a teacher interacts in the environment performing actions after receiving stimuli from the environment in which he/she is inserted into, thus generating a new observed case, as detailed in [27]. The actions taken by this teacher are observed by the system, which memorizes what the teacher has taught and stores it into the case base.

Observational learning is a passive learning technique, where the expert may not even know that he/she is being observed. A limitation of this technique is that it depends on the behavior of the observed specialist. For example, if during the observation the specialist does not have to solve a specific problem, then experiences (cases) will not be created to solve that problem. Even if the specialist is observed for a long period of time, or on multiple occasions, there is no guarantee that a complete sample of the problem space can be collected [10]. Likewise, if a large number of one type of game action is shown while a smaller number of another type is presented, this can lead to an unbalanced case base. For example, [28] used the observation learning technique to teach a robot to play football. By observing how other players acted, a case base that covered many cases of one action and few others were produced. In particular, 67.8% of cases involved running, 32.1% changing direction, and 0.1% kicking. In short, this demonstrates that a passive observation behavior did not allow capturing gaming experiences for rarer situations, which may need to be better observed in order to improve the performance of the system.

Using a system that monitors the actions of a teacher, it is possible to develop a case base capable of solving a large number of the most frequent problems occurring in a game environment. However, it is essential to note it is tiring for the instructor to perform all possible situations that can occur at specific times. To avoid this, techniques are needed to allow agents to learn from a few demonstrations provided by a teacher or to detect that there is no knowledge on how to solve certain situations in the case base, as tried out in this article. With this, "active learning" techniques [10, 11] can be explored in a complementary way to observation learning techniques. In practice, only when it is identified that a current case base has not the competence to solve a current problem, the specialist is invited to present a solution to this new problem, leading to the retention of a new case in the case base.

Aiming to decrease a teacher's demonstration effort, an example of active learning is the SALT (Selective Active Learning from Traces) technique [11]. With this, there is a Dl number of possible states that an intelligent system has learned and Dt is the total number of necessary cases that the system can encounter during its execution, where Dt can be very different from Dl. If the system finds a known situation, it acts alone. However, if a situation is found outside the scope of knowledge of the system, control is passed on to the teacher to solve the problem. This control remains with the teacher until an execution state is found in which the system is able to take control again. In practice, this active learning algorithm has been evaluated in two different game styles: a Super Mario platform game in which the player must reach the end of the level while avoiding enemies, and a puzzle game called Thermometers in which it is necessary to fill thermometers of different sizes and orientations with mercury where the amount of mercury is indicated by the line and column in which it is located.

C. *The Game of Truco*

The Truco [6] addressed in this work consists of the Truco "Gaudério", where this work is more specifically focused on matches involving the dispute between two players. Truco is based on the 40 cards of the Spanish deck discarding all eight, nine, ten, and wild cards. Like other card games, Truco is also susceptible to luck through the drawing of cards. A match is counted to 24 points, winning the first player to achieve this score. Each game is played in "hands" that are worth *n* points. To play a hand, each player receives three cards, where a hand can be divided into two main stages: ENVIDO and TRUCO. Although these disputes are different in the game, decisions, and information about cards possibly revealed by the players in the ENVIDO can influence the plays that are carried out in the TRUCO. At each stage, players have different ways of increasing the number of points that are played in the hand. There is also a special case of ENVIDO called FLOR, where it is necessary to have three cards of the same suit, thus scoring three points in case of victory. It is possible to earn a higher number of points in the game with the use of FALTA-ENVIDO and REAL-ENVIDO kinds of ENVIDO bets and with CONTRA-FLOR and CONTRA-FLOR-E-O-RESTO kinds of FLOR bets. Each hand can be played on the best of up to three rounds, in which the player who plays the highest card wins in individual round disputes. In each hand in the game, the order of whoever plays first alternates between the players. In this case, there are different ways to play (choice of moves), because whoever plays first (called as hand player) can play differently from who responds to a move (called as feet player). In a single-handed contest, the alternation between players is guided by the outcome of the hand's rounds. The player who wins a round must play first in the next round of that hand.

In the ENVIDO stage, if a player has two cards of the same suit he/she has ENVIDO points. To calculate these points, the value of the two cards of the same suit is added and to this value is added twenty (this is a fixed number due to having two cards of the same suit). The value of each card is its card number (a number from 1 to 7). A few cards (10 to 12) are not worth extra points in the ENVIDO. For example, if a player has a 6 of clubs and a 10 of clubs it means that he/she has 26 points (20 fixed points plus 6 points). From these numbers of points and cards revealed during the game, a player can make plausible inferences regarding the possible cards that the opponent possesses.

After the ENVIDO, there is the TRUCO stage, where bets can be proposed and increased at any time of the game by any player. When making a TRUCO bet, it is possible to earn a higher number of points in the game. Instead of earning 1 point for winning the hand, it is possible to earn up to 4 points through a so-called VALE-QUATRO bet. If a player makes a TRUCO bet, the opponent has three answer options. 1. To deny, it means that he/she folds that hand. Therefore, a new hand can be started since the player who made the TRUCO bet receives 1 point in the game. 2. To accept the TRUCO bet. If accepted, the current hand is worth two points in the game. 3. To raise, through RETRUCO, to make the bet worth 3 points. Likewise, a RETRUCO bet can be increased to VALE-QUATRO (worth four) to raise the bet to 4 points. As in the TRUCO stage, the player who wins 2, of 3, rounds wins the disputed hand.

As the game of Truco has different states of dispute, different game situations can be analyzed to guide the player's decision-making. In these situations, it is also possible to bluff in various ways in Truco, making ENVIDO and TRUCO bets without having a strong hand. If the opponent denies the bet, the player wins the points disputed in that hand. Players choose how to play (choose moves) according to everything that is known or can be inferred at a given time of the game, although there is always uncertainty due to incomplete information. Furthermore, the forms of play explored in these disputes can be dynamically changed, where initial decisions on how to play can be modified even during the different rounds of the same hand in the game. That is, the player may dynamically have to adapt to the situation and the opponent. In our view, this scenario leads to a case where decisions are made on a case-base basis. In AI, it means that CBR is a relevant approach to be explored in the modeling and development of agents to play the Truco game.

III. A CBR Approach to Play Truco

TrucoGame is a CBR system implemented in Java as part of the activities of the AI and Games research group in which this work is inserted into. In the TrucoGame, the case base is stored in a MYSQL database. To collect games of Truco, this system allows two human players to dispute matches over the web. Through it, complete Truco matches played among human players can be collected. These game playing logs contain all game moves advanced in those matches. There, each played hand was represented as a case in the case base. Cases represent various kinds of game information as presented in Table I. Because the Truco cards have different levels of importance in the game, the relevance of each card was represented through a nonlinear numerical scale (see Table II), and then this scale was used in the recording of the disputed hands of Truco.

To build the initial TrucoGame case base, called BASELINE in this work, the "learning by observation" technique was explored. Through observation of game matches, this case base recorded 48 matches played among pairs of human players. This resulted in a case base containing 513 Truco hands, which were recorded as individual cases. To perform the observations, the TrucoGame only observed the game actions played by one of the human players, randomly selected during the development of each match. In this way, the system memorized only what this player could observe in the game. For example, if a player withheld the cards (the cards were played face down on the table), such cards were not recorded in the case structure. That is because the human player being observed would not have access to that game information at the end of the match.

In this project, a single case base was initially used to provide answers for all types of CBR queries emitted by the implemented Truco player agents. However, with the growth of this case base (since cases were continually retained in it), the similarity calculations were a bit time-consuming. With the use of such a lazy learning approach (i.e., there is no generalization process as in other AI techniques), the similarity of the query against all past cases stored in the case base had to be computed to support each one of the agent moves in the current disputed matches. To improve the query response time, the initially collected case base was divided into 3 distinct case bases. In this way, each case base was directed to the resolution of a type of problem in the Truco game. Each case base started with the observed 513 Truco hands. Subsequently, as a result of the case learning methods detailed in this article, these case bases ended storing a number of independent cases (see Table IV). In addition, although the FLOR bet can be understood as an ENVIDO kind of game move in Truco, this type of bet is uncommon in the game. Therefore, an independent case base containing FLOR cases was also developed in this project.

Table I - Examples of attributes used in the representation of the Truco's hands

Kinds of attributes	Description	Local similarity function
Received cards	Cards received by players are represented as numeric attributes. The codification is nonlinear and ranges from 1 to 52. Following the Truco rules in [6], these numeric values capture the importance of the cards in the game (Table II). Then, the received cards are organized in low, medium, and high attributes.	Absolute value of the difference
Players' order	Player 1, or Player 2.	Equal
Cards played in each round of a hand	They capture the player's choices according to the strength of the cards (Table II).	Absolute value of the difference
Winner in each round of a hand	Player 1 or Player 2.	Equal
Bets made by the players	The ENVIDO and TRUCO bets made by each player.	Equal
Number of points available in the hand	Total points for ENVIDO bets.	Absolute value of the difference
Number of points won/lost	The number of points won or lost in each different game action executed in the hand, and the game score before and after the dispute of the hand.	Absolute value of the difference

Table II – Codification of the cards in the Truco cases

Category	Truco cards	TRUCO enconding	ENVIDO encoding
Top high	Ace of spades ♠	52	1
	Ace of clubs ♣	50	1
High	7 of spades ♠	42	7
	7 of diamonds ♦	40	7
High white	All 3's	24	3
	All 2's	16	2
Medium white	Ace of hearts ♥ and ace of diamonds ♦	12	1
Lower black	All 12's	8	0
	All 11's	7	0
	All 10's	6	0
Lower white	7 of clubs ♣ and 7 of hearts ♥	4	7
	All 6's	3	6
	All 5's	2	5
	All 4's	1	4

A. Case retrieval and game action reuse

Different kinds of queries are performed to support the agent's decisions in the TrucoGame. These queries are related to each one of the problems to be solved in the game. As each query is performed in the stages and rounds of the hand dispute, the attributes used as parameters in these queries are naturally different.

An example of a query for the resolution of an ENVIDO problem can be described. This query captures a game situation where the agent is the hand player of the disputed hand. In this position, the agent makes an ENVIDO bet. The observed agent, which has 29 points of ENVIDO, should decide whether to accept this bet or not. To solve this problem, a list of cases is retrieved from the case base. Considering the most similar retrieved cases, a solution to this problem is reused from them, where information about which hand player advanced an ENVIDO bet and which one won this bet is analyzed. In such Truco hand, the feet player has not yet played its first card on the table. Therefore, the opponent is not yet aware of this card. Thus, the information representing the feet player's first card is discarded in the query formation. As a result of the query execution, the 10 most similar cases are retrieved from the TrucoGame case base, allowing the agent to reapply a past game action, whether it is an honest or deceptive (a bluff) game move, into the resolution of the current problem. Another example of retrieval is presented in Table III. This example concerns a query that is performed to decide which card should be played at a given hand situation. In this query, the data representing the current game situation is detailed in the first row of Table III. Once this query is executed, it is possible to determine the game action executed by the majority of the retrieved cases, aiming to reproduce this action on the current game situation. In the retrieved cases, the low card was not played in only 2 cases (cases 337 and 414). Therefore, the majority's decision involves playing the low card from the agent's hand on the table. As detailed in the query, such a low card in the current hand is the 4 of spades. As presented in these examples, similar game moves are determined and executed by the Truco player agents implemented in the TrucoGame.

Table III – Using the retrieved cases to decide what card to play

Case	Who is the hand player	High card	Medium card	Low card	First played card	Sim (%)
Query	Agent (1)	3♦ (24)	10♣ (6)	4♠ (1)		
306	Agent (1)	3♣ (24)	11♦ (7)	4♠ (1)	4♠ (1)	99.03
378	Agent (1)	3♣ (24)	10♦ (6)	4♥ (1)	4♥ (1)	98.79
164	Agent (1)	3♦ (24)	10♣ (6)	4♠ (1)	4♠ (1)	98.55
337	Agent (1)	3♥ (24)	6♥ (3)	5♣ (2)	6♥ (3)	98.55
186	Agent (1)	3♣ (24)	12♣ (8)	4♦ (1)	4♦ (1)	98.07
389	Agent (1)	3♠ (24)	4♠ (1)	4♥ (1)	4♠ (1)	98.07
286	Agent (1)	3♦ (24)	11♥ (7)	5♦ (2)	5♦ (2)	97.59
441	Agent (1)	3♠ (24)	11♦ (7)	4♥ (1)	4♥ (1)	97.59
213	Agent (1)	3♣ (24)	10♣ (6)	6♥ (3)	6♥ (3)	97.11
414	Agent (1)	3♠ (24)	7♣ (4)	6♠ (3)	7♣ (4)	96.75

The TrucoGame uses the "majority voting" criteria in its reuse method. It means that the most common decision in the retrieved cases for a given query is reused in the resolution of the current problem situation. In practice, the K cases retrieved for a query vote on their game actions. Importantly, the most common game action advanced in the past is only reused if it either resulted in a hand victory or resulted in the loss of the least number of points.

IV. CBR Learning via Case Retention

One of the key knowledge repositories [3] of a CBR system is the case base. To support the construction of competent case bases to be used by Truco player agents, alternative case retention techniques were investigated in this project. As part of training Truco games, agents played against a version of themselves taken as opponents. In this self-play learning model, the opponent agent only used the BASELINE case base to support its game moves. This BASELINE case base never retained new cases as a result of such case retention experiments. It remained to store the initially collected 513 cases.

Initially, this work explored the ACE technique. In doing so, the BASELINE case base was used in the computation of responses to CBR queries. However, when a query has not returned at least 10 cases with similarity values greater than 90%, the system executed a random game move as described in Algorithm 1. This learning strategy is explored in order to prevent the system from always reproducing the same past game actions, without learning something new, since the BASELINE case base stored a limited number of cases.

Algorithm 1 – Reuse a past game action or randomly play

```
Algorithm: reusePastOrPlayRandomGameAction

1. data: CB, a case base ∈ {envidoCB, florCBR, trucoCB}; query, a query case regarding the current problem; simThreshold, a similarity threshold; numCases, a minimal number of retrieved cases; Res, a list of retrieved cases ranked by similarity; aReuseCriteria, majority voting reuse criteria;
2. result: aGameAction, a selected game action.
3. begin
4.   Res = {case_1, case_2, …, case_10 ∈ CB | (sim(query, case_n) >= simThreshold)}
5.   if |Res| >= numCases then
6.     aGameAction = reuseGameAction(Res, aReuseCriteria)
7.   else
8.     aGameAction = randomGameAction(query)
9.   end if
10.  return aGameAction
11. end
```

To decide when to retain a new case in the case base is one of the main problems of case base maintenance tasks in CBR systems [13, 14]. As the time required by the similarity computations and the consequent retrieval of cases from the case base is directly related to the size of the case base, if the used case base stores redundant cases, for instance, the response of performed queries may not be efficient. To approach this problem, the "case substitution" technique was also investigated in this project. In summary, to represent these new cases in the case base, two types of retention techniques were approached in the Truco card game: "retention of new cases" and "retention of new cases with substitution.

A. *The retention of cases in the TrucoGame case base*

Similar to the learning strategy described in [5], to limit the size of the case base (consequently, reducing the response time of queries) and to avoid the retention of a large number of cases in the case base, a new case was added into this case base only when the performed query has not retrieved cases with a similarity value greater than the defined threshold. Because each decision in the TrucoGame is the result of a voting method where the 10 most similar retrieved cases place a vote on their game actions, a new case is retained into the case base if fewer than 10 cases with similarity greater than the threshold value are retrieved from the case base (Algorithm 2).

Algorithm 2 – Retention of new cases

```
Algorithm: retainNewCase

data: CB, a case base ∈ {envidoCB, florCBR,
  trucoCB}; query, a query case regarding the
  current problem; curCase, current case
  description and solution; simThreshold,
  similarity threshold; numCases, minimal
  number of retrieved cases; Res, a list of
  retrieved cases ranked by similarity;
result: resCB, a resulting case base.
begin
  Res = {$case_1$, $case_2$, …, $case_{10}$ ∈ CB |
    (sim(query, $case_n$) >= simThreshold)}
  if |Res| < numCases then
    resCB = CB + curCase
  else
    resCB = CB
  end if
  return resCB
end
```

Algorithm 3 – Retention of new cases with substitution

```
Algorithm: retainNewCaseWithSubstitution

data: CB, a case base ∈ {envidoCB, florCBR,
  trucoCB}; query, a query case regarding the
  current problem; curCase, current case
  description and solution; simThreshold,
  similarity threshold; numCases, minimal
  number of retrieved cases; Res, a list of
  retrieved cases ranked by similarity;
  pastCase: past case retrieved for a given
  query;
result: resCB, a resulting case base.
begin
  Res = {$case_1$, $case_2$, …, $case_{10}$ ∈ CB |
    (sim(query, $case_n$) >= simThreshold)}
  if |Res| < numCases then
    resCB = CB + curCase
   else
      foreach pastCase in Res do
         if curCase.score > pastCase.score
         then
               resCB = CB – pastCase
         end if
      end foreach
      resCB = CB + curCase
   end if
   return resCB
 end
```

The "retention of new cases with substitution" is a variation of the "retention of new cases" technique. In it, even when a case used as a query is not considered new since the case problem is highly similar to problem-solving experiences captured in cases already stored in the case base, there may exist a case stored in the case base that is considered "worse" than the current case used as query. When it happens, the past case is replaced by the new case. In practice, each new case was compared to the retrieved cases from the case bases, allowing the use of the overall game score obtained at that stage of the game to assess whether the new case is important to be retained. In this situation, a case considered relevant for the TRUCO stage, and not so relevant for the ENVIDO stage, is memorized only in the case base in which it has demonstrated to permit to obtain a better game result. This better result evaluation is measured by the number of points that the game actions stored in the cases earned in the game. In this retention strategy, therefore, a new case is memorized only when it permits to obtain a relevant score in the game hand (Algorithm 3).

Because there may be situations in Truco where it is not possible to make a game move that results in the earning of many points, one game move is also considered better than another one when it results in a lower loss of points. During a Truco match, for example, where the player accepted an ENVIDO bet and ended up losing two points to its opponent, the action of accepting this ENVIDO bet is considered a worse move than the action of declining from it. That is because the bet refusal resulted in the loss of only one point to the opponent.

Algorithm 4 – Asking for the help of the human player

```
Algorithm: askHumanPlayerHelp

data: CB, a case base ∈ {envidoCB, florCBR,
  trucoCB}; query, a query case regarding the
  current problem; simThreshold, a similarity
  threshold; numCases, a minimal number of
  retrieved cases; Res, a list of retrieved
  cases ranked by similarity; aReuseCriteria,
  majority voting reuse criteria;
result: aGameAction, a selected game action.
begin
  Res = {$case_1$, $case_2$, …, $case_{10}$ ∈ CB |
    (sim(query, $case_n$) >= simThreshold)}
  if |Res| >= numCases then
    aGameAction = reuseGameAction(Res,
    aReuseCriteria)
  else
    aGameAction = askForHelp(query)
  end if
  return aGameAction
end
```

To demonstrate how the "retention of new cases with substitution" works, an example of an ENVIDO query where such retention occurred can be presented. At the end of the disputed hand, a query on the ENVIDO case base returned the 10 most similar past cases. Since all retrieved cases have a similarity greater than the defined threshold (90%), it is analyzed whether there is any retrieved case showing a result that is worse than the currently observed result. Of these, case 243, for example, had a worse result than the result of the new case. In this situation, the case

243 resulted in a balance of –1 point (the opponent won 1 point), while the new case resulted in a balance of zero points. So, the retrieved case that is worse than the current one (a defeat) is removed from the case base, being replaced by the new case.

There are situations in the Truco game in which the agent may encounter an unknown problem situation, where the repetition of the past game move from the retrieved cases may not provide a good solution for the current problem. Furthermore, the execution of random game action, as in the "automatic case elicitation" technique, may not lead to the development of a valid/coherent strategy in the game. To overcome these problems, another approach for the retention of new cases in the case base investigated in this work involves the "active learning" technique [10, 11].

In the active learning context, a CBR query is executed. As a result, there may not exist past cases stored in the case base that have a similarity greater than 90% with the current game situation captured in the query. When the case base doesn't contain problem-solving experiences sufficiently similar to the current problem, the system asks for the assistance of a human player (Algorithm 4). In doing so, the system presents to the human player all the game information that is known at the moment that the query is formed. That is, the system shows to the human player the visible state of the game according to the perspective of the agent that has to make a game move. With that, the human player can indicate which game action should be executed. To do this, a decision function is used to identify which game situations the agent has the knowledge to solve and which ones it needs to ask for the help of the human player. As a result of this case learning process, a new problem-solving experience can be formed and stored as a case into the case base.

B. *The exploration of case retention approaches*

To assess the effectiveness of such learning techniques in the context of the Truco card game, the case base containing the 513 Truco hands from the 48 matches played between pairs of human players (a mixture between novice and experienced players) was initially used to guide the game decisions of the implemented agents (here called BOT). As described earlier, this initial case base was called BASELINE. Then, alternative case retention techniques aimed at improving the competence of this BASELINE case base were tested in this project. To approach this problem, a set of 300 training Truco matches was played, where different Truco player agent configurations were explored in these "training" matches.

During this learning process, which focused on the case retention, 5 different agent setups were used in the disputed matches. There, a type of BOT played against another type. Each one of these 5 agents relied on its case base for the answering of its queries, where such case bases recorded the result of the case retention processes performed in the 300 matches. While one of the agents was learning with the development of these training games, the opposing one had always used the BASELINE case base to determine its game actions. This initial agent configuration is called BOT0. Starting from the BASELINE case base, the case bases used by each one of these 5 agents were continuously expanded and modified during the dispute of these training games, while the BASELINE case base used by BOT0 was kept unmodified. In summary, each one of these 5 different agent configurations played 300 independent matches against the BOT0 as part of the training Truco games. The case retention settings used by the 5 different agents are the following:

BOT1 - Retention of new cases: each game move is determined as a result of a voting strategy as implemented in the TrucoGame, with the voting of each one of the 10 most similar cases retrieved for a query executed by BOT1 in its case bases. With that, the new case is only retained in the BOT1 case base if its query doesn't retrieve at least 10 past cases with similarity greater than 90%;

BOT2 - Retention of new cases with substitution: as a result of a query, if it is not possible to retrieve from the BOT2 case bases at least 10 cases with similarity greater than 90%, the case problem captured by the query is considered as new. From this, this case is retained in the BOT2 case bases. However, if 10 similar cases are retrieved, each one of these 10 retrieved cases is compared with the query. If there is a retrieved case with a game score (i.e. number of points earned in that disputed hand) that is lower than the score of the current case, the past case retained in the BOT2 case base is not considered as good as the current one. In this way, the case retrieved by the query is removed from the BOT2 case base, and the new case is inserted into this case base.

BOT3 - Random play and retention of new cases: for each performed query, when it is not possible to retrieve at least 10 cases with similarity greater than 90%, this agent plays a random game move. In the developed implementations, however, this random move only involves the acceptance/refusal of TRUCO and ENVIDO bets. In this situation, decisions regarding which cards to play and in what order to play them remain being reused from the cases retrieved as a result of the executed query. Similar to the retention strategy explored by BOT1, BOT3 performs the retention of new cases in its case bases.

BOT4 - Random play and retention of best cases: it combines the techniques used by BOT2 and BOT3. Similar to BOT3, BOT4 retains cases in its case base resulting from random moves. Similar to BOT2, the retention technique used by BOT4 also removes cases returned for queries in which the number of points earned is lower than the number of points earned at the current case situation.

BOT5 - Active learning case retention: the active learning approach was used in BOT5 tutoring. In each BOT5 game move, if it is not possible to retrieve from the BOT5 case base at least 10 cases with similarity greater than 90%, BOT5 requests the help of the human player (the first author of this paper, with more than 10 years of Truco experience). The aim is to determine what game action has to be executed in the current situation. When requesting such help, BOT5 presents to the human player all the game information that this agent would normally use in its query. Therefore, the human player can evaluate this information to decide the next game move to be made by BOT5 in that hand situation. All game actions performed by this BOT5 and their results in the disputed match are recorded in the structure of the new case. In the end, this new case is only retained into the BOT5 case base if at least one game move has been demonstrated by the human player.

All BOT1 to BOT4 played against BOT0 in this case learning process. Each competition took around 12h to perform the 300 training Truco matches. For BOT5, the presence of a human player was required during the development of these 300 matches. During the BOT5 tutoring, the number of human player' requests for help were quite constant, especially in relation to the TRUCO bet decisions, which are the game actions that consider the highest number of attributes in the formed queries. Even with the 300 training matches, it was not possible to observe some rare Truco situations such as the occurrence of CONTRA-FLOR bets. While BOT5 was being trained, for example, only 6 hands involved CONTRA-FLOR situations, in a total of ~6000 played hands. This demonstrates that there may be necessary to carry out such learning in a "guided" manner (so far not explored in this project), where cards would be drawn in such a way that these rare game situations could more frequently appear during training.

V. Experiments and Results

The case retention learning involved each one of the 5 agent configurations disputing the 300 Truco matches against the BOT0. During these matches, each hand was generated with shuffle and distribution of cards at random, allowing the BOT to record different game problem-solving experiences in their case bases. After this learning stage, approximately 6,000 played hands were obtained, where case bases of different sizes were constructed as presented in Table IV. With the exception of the BOT5 case base, which is the result of the active learning process, the size of the other case bases is similar. As far as possible, the tutor sought to perform different playing styles as part of the BOT5 tutoring games.

In the experiments carried out in this work, it was explored the test methodology of the Annual Computer Poker Competition (ACPC) [29]. According to it, a duplicate match approach was used, where all players play with the same sets of drawn cards. For example, in each hand of a match the agent receives the set of cards directed to its position on the table. At the end of the match, the agents exchange positions and re-play the match using the previously dealt cards. This strategy allows the reduction of the variability of the cards and the adjustment of the quality of the cards received by the players.

Table IV – Number of cases in each constructed case base

Case base (CB)	Played hands	envidoCB	florCB	trucoCB
BOT0 CB	513	513	513	513
BOT1 CB	6,859	1,183	689	1,401
BOT2 CB	6,338	1,656	704	1,445
BOT3 CB	6,263	1,171	688	1,711
BOT4 CB	5,876	1,497	698	1,717
BOT5 CB	6,562	681	520	5,125

To evaluate the results of the tested case retention strategies, the 5 different agents using their respective case bases played 30 matches against the BOT0. This number of test matches is equivalent to 10% of the total matches used in the training phase. This resulted in a total of 60 matches in which the agents altered their positions on the table.

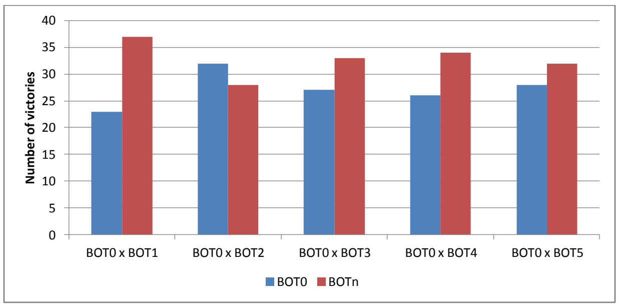

Fig. 1 – Results from the testing Truco matches

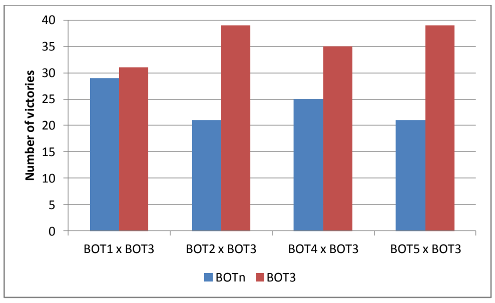

Fig. 2 – Results from the BOT3 Truco matches

Fig. 1 shows that BOT1 achieved the best results in the tests. From BOT1 through BOT4, BOT1 may be considered the "most human" one because it only reuses previously collected human player game moves. This is different from BOT2, which deleted cases from its case bases as part of the case retention learning, and BOT3, and BOT4, which randomly played and stored cases in their case bases. BOT5, which used the case base resulting from the tutoring process, presented an unexpected outcome in the tests. In general, there was an initial expectation that BOT5 would have an advantage over the others since its case bases were the result of the direct intervention of the human player. BOT5 also had a much larger TRUCO case base. Even considering that, BOT5 has not achieved a large number of victories against the BOT0. BOT2, which performed the case substitution strategy in the formation of its case bases, was the only one that lost against the BOT0. This may be due to the fact that BOT2 forgot some of its cases, which were removed from its case base because they reflected a larger loss of points when compared to game moves that avoided risk bets (accepting or increasing the number of points in the dispute). For example, BOT2 proposed a VALE-QUATRO bet having strong Truco cards in its hand. Despite this positive situation, luck factors lead BOT2 to lose that past hand dispute. In the retention of new cases with substitution, this case situation, previously stored in the BOT2 case base, ended being forgotten. Even if the BOT2 had received these high cards, and the chance of winning was high, the fact that that agent had lost eventually led the BOT2 to forget this case situation during the case retention learning.

To determine which one of these tested agents was the best Truco player, along with the assessment of the case learning strategies tested in this work, Truco matches played among the different BOT were also disputed. Such competition used the same set of cards that were explored in the matches played against the BOT0. As a result, BOT3 (which memorized new cases and made random game moves whenever necessary) achieved the best results (Fig. 2). BOT3 overcome all other agents with a reasonable margin of victories, with the exception of BOT1 that explored a similar case retention strategy to form its case base.

VI. Conclusions

Card games are great testing and development environments for new AI techniques. One of these techniques is CBR, which has been under-investigated in the development of agents for new kinds of games, as it is the case of the Truco game. However, CBR agents rely on competent case bases, making it difficult (if not impossible) to develop an efficient agent for new kinds of games when these case bases are not promptly available for various reasons. To approach this problem, alternative case learning strategies focusing on the improvement of the initially collected case bases are investigated in this work. With the learning techniques implemented and tested, the resulting agents were able to play better when compared to agents based on a case base initially collected with human players. The exception of this was the active learning technique since the broad manner in which it was explored in this work has not lead to a better agent. Further research in this direction is under-development in our AI and Games research group with particular attention to the capacity of actively teaching the agents to buff in the Truco game. In addition to evaluating the use of other case learning methods and developing other kinds of validation experiments (including disputes between agents and human players), future works can investigate these learning approaches along with other solution reuse strategies [7] (other than the majority solution criteria) as described in other works developed in our AI and Games research group.

References

[1] J. Hamari and M. Sjöblom, "What is eSports and why do people watch it?," *Internet research,* vol. 27, pp. 211-232, 2017.

[2] S. Ontanón and A. Ram, "Case-based reasoning and user-generated artificial intelligence for real-time strategy games," in *Artificial Intelligence for Computer Games*, ed: Springer, 2011, pp. 103-124.

[3] R. L. De Mantaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M. L. Maher, M. T COX, and K. Forbus, "Retrieval, reuse, revision and retention in case-based reasoning," *The Knowledge Engineering Review,* vol. 20, pp. 215-240, 2005.

[4] D. W. Aha, M. Molineaux, and M. Ponsen, "Learning to win: Case-based plan selection in a real-time strategy game," presented at the Int. Conf. on Case-Based Reasoning (ICCBR-05), Chicago, IL, USA, 2005.

[5] V. Andreeva, J. Beland, S. Gaudreau, M. W. Floyd, and B. Esfandiari, "Creating Non-Player Characters in a First-Person Shooter Game Using Learning by Observation," presented at the Workshop on Case-Based Agents, The 22nd Int. Conf. on Case-Based Reasoning (ICCBR 2014), Cork, Ireland, 2014.

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

[7] G. B. Paulus, J. V. C. Assuncao, and L. A. d. L. Silva, "Cases and Clusters in Reuse Policies for Decision-Making in Card Games," presented at the IEEE 31st Int. Conf. on Tools with Artificial Intelligence (ICTAI 2019), Portland, OR, USA, 2019.

[8] G. Castillo, S. Avendaño, and N. A. Goussies, "An Human-Computer Interface Using Facial Gestures for the Game of Truco," presented at the Iberoamerican Congress on Pattern Recognition, Buenos Aires, Argentina, 2012.

[9] M. Fagan and P. Cunningham, "Case-based plan recognition in computer games," presented at the Int. Conf. on Case-Based Reasoning (ICCBR 2003), Trondheim, Norway, 2003.

[10] M. W. Floyd and B. Esfandiari, "An active approach to automatic case generation," presented at the The 8th Int. Conf. on Case-Based Reasoning (ICCBR 2009), Seattle, WA, USA, 2009.

[11] B. Packard and S. Ontanon, "Policies for active learning from demonstration," presented at the 2017 AAAI Spring Symposium Series, Stanford University, 2017.

[12] J. Schaeffer and J. van den Herikb, "Games, computers, and artificial intelligence," *Chips Challenging Champions: Games, Computers and Artificial Intelligence,* p. 3, 2002.

[13] B. Smyth and E. McKenna, "Competence models and the maintenance problem," *Computational Intelligence,* vol. 17, pp. 235–249, 2001.

[14] J. M. Juarez, S. Craw, J. R. Lopez-Delgado, and M. Campos, "Maintenance of Case Bases: Current Algorithms after Fifty Years," presented at the Twenty-Seventh Int. Joint Conf. on Artificial Intelligence (IJCAI-18), Stockholm, Sweden, 2018.

[15] S. Ontañón, "Case acquisition strategies for case-based reasoning in real-time strategy games," presented at the The Twenty-Fifth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2012), Marco Island, Florida, USA, 2012.

[16] G. M. Farouk, I. F. Moawad, and M. M. Aref, "A machine learning based system for mostly automating opponent modeling in real-time strategy games," presented at the The 12th Int. Conf. on Computer Engineering and Systems (ICCES 2017), Cairo, Egypt, 2017.

[17] M. Certický and M. Certický, "Case-based reasoning for army compositions in real-time strategy games," presented at the 13th Scientific Conference of Young Researchers (SCYR 2013), 2013.

[18] D. S. L. Ariza, A. A. Sánchez-Ruiz, and P. A. González-Calero, "Time Series and Case-Based Reasoning for an Intelligent Tetris Game," presented at the Int. Conf. on Case-Based Reasoning (ICCBR 2017), Trondheim, Norway, 2017.

[19] J. H. Powell, B. M. Hauff, and J. D. Hastings, "Utilizing case-based reasoning and automatic case elicitation to develop a self-taught knowledgeable agent," presented at the Challenges in Game Artificial Intelligence: Papers from the AAAI Workshop, 2004.

[20] J. H. Powell, B. M. Hauff, and J. D. Hastings, "Evaluating the effectiveness of exploration and accumulated experience in automatic case elicitation," in *The 6th Int. Conf. on Case-Based Reasoning (ICCBR 2005)*, Chicago, IL, USA, 2005, pp. 397-407.

[21] I. Watson and J. Rubin, "Casper: A case-based poker-bot," in *Australasian Joint Conference on Artificial Intelligence*, 2008, pp. 594-600.

[22] J. Rubin and I. Watson, "Similarity-based retrieval and solution re-use policies in the game of Texas Hold'em," presented at the Int. Conf. on Case-Based Reasoning (ICCBR 2010), Alessandria, Italy, 2010.

[23] J. Rubin and I. Watson, "Case-based strategies in computer poker," *AI communications,* vol. 25, pp. 19-48, 2012.

[24] A. Sandven and B. Tessem, "A case-based learner for poker," presented at the The Ninth Scandinavian Conference on Artificial Intelligence (SCAI 2006), Helsinki, Finland, 2006.

[25] M. M. Richter and R. O. Weber, *Case-based reasoning*: Springer, 2016.

[26] J. H. Powell and J. D. Hastings, "An empirical evaluation of automated knowledge discovery in a complex domain," presented at the Workshop on Heuristic Search, Memory Based Heuristics and their Applications: Twenty-First National Conference on Artificial Intelligence, AAAI-06, Boston, Massachusetts, USA, 2006.

[27] M. W. Floyd and B. Esfandiari, "A case-based reasoning framework for developing agents using learning by observation," presented at the 23rd IEEE Int. Conf. on Tools with Artificial Intelligence (ICTAI 2011), Boca Raton, FL, USA, 2011.

[28] M. W. Floyd, B. Esfandiari, and K. Lam, "A Case-based Reasoning Approach to Imitating RoboCup Players," presented at the The Twenty-First International FLAIRS Conference (2008), Coconut Grove, Florida, USA, 2008.

[29] ACPC. *Annual Computer Poker Competition*. Available: http://www.computerpokercompetition.org/