Investigating Cases and Clusters-Based Reuse Policies for Card-Playing Agents

Gustavo B. Paulus
Federal Institute of Education, Science and Technology of Rio Grande do Sul
Ibirubá, Brazil
gustavobpaulus@gmail.com

Daniel P. Vargas
Graduate Program in Computer Science
Federal University of Santa Maria
Santa Maria, Brazil
pinheiro.vargas@gmail.com

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

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

***Abstract*—Computer games continue to present challenging experimental fields for developing Artificial Intelligence (AI) models. With Case-Based Reasoning and Clustering, this work proposes novel cases and clusters-based reuse criteria for implementing card-playing agents with diversified playing skills. Using the game of Truco, a common game in South America, we detail how game actions are reused from past cases selected as query answers for given game problems. In doing so, the majority rule, the probability-based lottery, the probability of victory, and the number of points won reuse policies are used to select a cluster of cases. Then these policies are also used to select game actions from the cases within the selected cluster. Investigating the combined exploration of reuse policies, experiments of different natures evaluate the performance of implemented Truco bots disputing matches against each other. Players in these tournaments are equipped with varied policies and use case bases constructed differently.**

***Index Terms*—Case-Based Reasoning, Clustering, Reuse Policies, Games, Truco.**

I. Introduction

The partial visibility of the game, the stochastic nature, and the possibilities of alternative kinds of bets present a relevant field for investigating novel card-playing game-AI techniques. To approach decision-making problems in these games, this paper explores records with concrete card-playing instances and other in-game logs where card-playing patterns can be analyzed and reused. There, data and knowledge are collected from disputes between players with different game-playing skills to guide decision-making in new problems. The proposals of this work are grounded on the recording and reuse of game-playing experiences via Case-Based Reasoning (CBR) [1]. Then, Clustering [2] is used to analyze and organize these experiences in groups. This work also investigates a card-playing application for AI called Truco [3].

Truco and Poker have similarities (e.g., card sorting, luck, rounds of dispute, and a few others) that warrant the exploration of past Poker game-AI proposals. In [4]–[6], logs of played Poker hands are represented in the format of cases maintained in case bases to support the decision-making in new problems. According to the work of [7]–[9], case-based decision "reuse policies" have a critical role in selecting and reusing game actions recorded in the cases selected as answers to CBR queries. As the content of card-playing cases can be analyzed in different ways, this paper shows that existing reuse policies [5], [6], namely the majority rule (MJ), probability-based lottery (PL), probability of victory (PV), and number of points won (NP) (i.e., the best outcome), can be enhanced to better consider the different decision-making states in which the game actions are taken.

The contribution of this work is to propose novel cases and clusters-based decision reuse policies to equip card-playing agents with varied game-playing skills. In the resulting approach, CBR represents cases to compute decisions for new problems, and clustering identifies distinct problem situations in the game. Clustering is used to analyze decision-making scenarios and consequent decisions encountered in the game-playing logs recorded as cases in case bases. The paper investigates the various hierarchical combinations of the MJ, PL, PV, and NP reuse policies while approaching the decision-making needs of Truco agents. It also explores self-play case learning results where various combined policies are implemented by agents involved in Truco tournaments. Experiments with alternative Truco bots are conducted, and the main results show that the case and cluster reuse criteria improved the assessment of game actions to be reused by the tested agents. Game actions and their different game states could be better distinguished when exploring a decision-making process where first clusters and then cases are reused. The effectiveness of standard reuse policies was also improved due to the use of large case bases resulting from the self-play method of case learning.

This paper is organized as follows: Section II reviews background information about CBR and Clustering, related

979-8-3503-2277-4/23/31.00

works mainly focusing on game-AI for Poker, and important Truco characteristics. Section III details the proposed approach through the case representation model for capturing Truco hands, the method of case retrieval and game action reuse, and mainly the cases and clusters reuse policies for game-playing decision-making. Section IV details experiments of different natures and then examines the obtained results. Section V briefly presents final remarks and future work.

II. Background to This Work

The work in [10] presents AI techniques to implement agent decision-making in card games. The work in [11] introduces a case acquisition strategy and new metrics to evaluate the agents' performance according to the similarity with experts' behaviors. In [12], CBR is explored in the opponents' learning and adaptation in the Real-Time Strategy (RTS) game GLest. [13] reviews CBR to approach problems in RTS games. [14] uses the "Automatic Case Elicitation" approach to form the case base. There, a random solution attempt is performed whenever a past case to solve the current problem is not found when computing a CBR query. Then the quality of this trial is assessed, and new cases can be created.

The authors in [15] describe the psychological and emotional aspects that make some card games an essential domain for advancing game AI. Like the Truco game, for example, the Bridge game, investigated in [16], has stages of decision-making leading to situations where different playing skills are required. Unlike the Poker Texas Hold'em [15], the game-playing actions most often require the ability to make bets. There, CBR reuse policy results are used to choose the bets that should be taken in the game. The solution (or game action) used by most retrieved cases usually reflects the choice of the solution commonly used in the past. However, the players/agents may be predictable if they consistently reproduce the most common game action. The work of [6] uses a probabilistic function to get around this problem.

Perhaps the AIs most similar to what we propose in this work is the CASPER system (CASe-based Poker playER), designed to play Texas Hold'em Poker [17]. CBR techniques decide the strategies to apply during each Poker match stage. They consist of actions to play given the game stage: raise, bet, fold, and take no action, among others. The agent can also play against alternative playing styles, mainly aggressive or passive. Like CASPER, SARTRE [6] uses CBR in Poker to make betting decisions based on past cases. Unlike CASPER, developed for the 10-player Texas Hold'em style, SARTRE is geared toward two players' matches, as in our work. On the grounds of alternative reuse criteria [5], [6], a CBR reuse model is proposed in [7], and a set of different kinds of case-based BOTs are evaluated in [8], [9]. In our work, variations of these bots were built using clustering as a second step in decision-making.

A. Case-Based Reasoning and Clustering to Card Games

CBR [1] is a lazy learning approach for game AI in which solutions successfully used in the past are reapplied to resolve new problems. Different reasoning tasks are performed by the CBR cycle of retrieval, reuse, revision, and retention. The retrieval is directed to similarity computation and determining the most similar cases to the query problem. The K-Nearest Neighbors algorithm [1] is often used to retrieve the most similar cases stored in the case base. Once a list containing the most similar cases is retrieved from the case base, reuse policies permit assessing the content of these cases to choose the solution to be reused. Reuse policies detailed in the game AI literature [5], [6] are:

- *Majority rule (MJ)*: the choice of game actions is supported by the solution recorded in the most similar retrieved cases;
- *Probability-based lottery (PL)*: the choice of game actions relies on the execution of a random function, where this function is based on probability estimates calculated from the solutions recorded in the retrieved cases. The higher the probability is, the higher the chance of choosing the game action;
- *Probability of victory (PV)*: the chances of victory regarding each different solution recorded in the retrieved cases guide the choice of game actions. The higher probability is chosen; and
- *Number of points won (NP)*: the number of earned points (i.e., the best outcome for the recorded solution) supports the choice of game actions.

Another important CBR task is the retention of new cases in the case base, especially to improve the case base competence to solve new types of problems [18]. To approach the retention problem, outstanding game-playing agents' performances have been achieved with the exploration of self-play learning [19]. For example, in [5], a Poker player CBR agent was built entirely from self-play matches.

Instead of only exploring CBR, this paper shows how to integrate CBR and Clustering to implement novel reuse policies for card-playing agents. The results of clustering algorithms add another layer for choosing a game action, making diverse choices, and often revealing (i.e., not losing) game-playing implicit information in the case base. It allows the identification of groups of existing cases, enhancing the retrieval, reuse, revision, and retention techniques. Furthermore, an embedded clustering can prevent misleading solutions from being used in recommendations [20] or to assist in the execution of activities regarding the maintenance of case bases [21].

Clustering is often used in CBR systems to index large case bases and, consequently, to reduce the number of similarity computations. These algorithms permit organizing sub-case bases according to the different types of problems. Although clustering results are often used to optimize the CBR query processing time, the clusters can also improve case-based recommendation accuracy. By allowing pattern recognition, clusters can reduce the data complexity, facilitating the identification of existing categories of in-game data. For instance, in [22], the clustering contribution increased the contextual data collection to game-playing behaviors and tactics captured

by game logs. Clustering is also used in games for finding strategies and players' profile groups (e.g., [23]). Then these groups can anticipate plays or adjust the game to the identified players' behaviors (e.g., [24]).

B. Truco: A Card Game for Applied AI

The game of Truco [3] involves cognitive challenges due to the partial view of the opponent's cards, the random card drawing, and the various possibilities of bluffing. Truco has several decision-making stages due to the multiple game interactions in the disputed hands. The 40 Truco cards (Fig. 1) are shuffled and distributed in each playing hand, with each player receiving three cards. Truco is a turn-tacking game where players take turns to play cards and place bets on the current hand. The winner of a turn starts the next game turn in that hand, and so on. The player must win the majority of the hand turns to win a Truco hand.

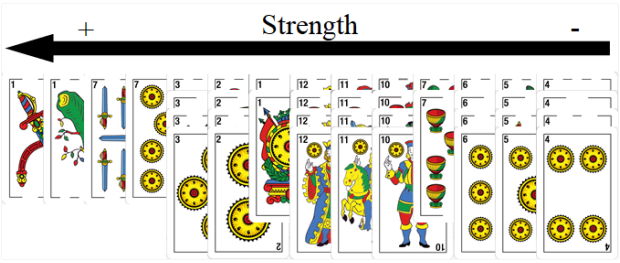

Fig. 1: The strength of the 40 Truco cards.

The main Truco game actions can be divided into "playing cards" and "placing bets". With a playing hand, players aim to drop a card higher than their opponent's on the three turns of a hand. Bets are placed to increase the points earned in the disputed game hands. Although bets are usually made to increase winnings when the player has strong cards, they can also be made to force the opponent to fold in the event of a bluff. A player has the option to score using two different bet modes, *Truco* (naming the game) and *Envido*. Players can accept, fold or raise the placed bets.

Similar to Poker, after each Truco card is played, the information available to guide the choice of the next game action is more complete. In the one-handed game (where two players challenge each other), the first player can receive 9,880 possible hands, while the second can receive 7,770. However, as many cards have the same value, the total number of equivalent game states drops to 502 hands (*Truco* mode) and 22 possible scores (*Envido* mode). The total amount of combinations grows significantly because the Truco players may place different bets during the game.

III. A CBR Approach to Play Truco

A TrucoGame system was developed as a result of our project. Through it, complete Truco matches played among human players were collected, recording the game moves in those matches. Based on the work of [7]–[9], each case is mainly represented as the following set of attribute-value pairs: "the received cards", "the players' order in the game", "the cards played in each turn of a hand", "the winner in each turn of a hand", "the bets made by the players", "the number of points won/lost", and "the number of points available in hand". Due to the Truco cards having different levels of importance in the game, the relevance of each card is represented through the non-linear numerical encoding and the Truco categories described in [8] and [3].

Learning by observation [25] was used to build the initial Truco case base called "*Extended Baseline Case Base (EBCB)*". The game actions played by one of the human players, randomly selected during the development of each match, were observed for the case representation. This case base contains 3,195 cases involving pairs of human players. Such cases reflect the actual behavior of the human players, with their strengths and weaknesses. Then the K-Means algorithm [2] (with the Euclidean distance function) was explored to create alternative indexing structures for the constructed case bases. The choice of the number of groups used as input on the execution of the clustering algorithm was performed using the WCSS (Elbow) technique [2]. The clusters formed with the analysis of Truco cases in our work are similar to the cluster example presented in [7].

A. Case Retrieval and Game Action Reuse

Decisions in Truco are derived from CBR query results. The parameters that form these queries vary in each stage and round of the hand dispute. The implemented case retrieval algorithm used a similarity threshold method to ensure that the most similar game-playing decisions for a given query are reused. Only these cases are used as input in the computation of the reuse criteria investigated in this work. With a ranked list of retrieved cases, the "majority voting" rule was used as our work's initial reuse method. It means the K cases retrieved place a vote on their game actions. Despite implementing the majority rule as its decision reuse policy, preliminary experimental results in [7] show that reuse policies can be improved when cases and clusters are used in the decision-making computations. Although that work analyzed the use of the combined use of the same reuse policy, this work significantly extends it and investigates distinct reuse criteria for the choice of clusters and solutions.

Algorithm 1 details how the two-step reuse model proposed in [7] works. The inputs of it are the cases stored in the case base, the list with the clusters where the most similar retrieved cases are organized (other clusters in which no retrieved case is present are not considered in the reuse computations), the query case that captures the current game state, the minimum similarity threshold value that each retrieved case should have, the reuse criterion to be used when selecting one of the clusters that are represented in the retrieved cases and the reuse criterion to be used when selecting the solution for the current problem. The reuse criterion is executed to select a cluster from the list of clusters in the most similar retrieved cases. As a result, only cases that belong to the selected cluster are kept in the list of retrieved cases. Only these cases are considered in the solution reuse computations. Thus the reuse criterion

selects the solution the agent should execute to respond to the current problem.

Algorithm 1 The computation of different reuse policies with a two-step reuse model.

Input: CB, case base; Clusters: clusters of cases; query, current game problem situation; simThreshold, similarity threshold; numCases, minimum number of retrieved cases; Res and Res', list of retrieved cases ranked by similarity; aSelectedCluster, selected cluster of cases; aReuseRule and aClusterReuseRule, reuse criteria $\in \{MJ, PL, PV, NP\}$.

Output: aGameAction, selected game action.

```
1: aGameAction = NULL
2: Res = {{case_1, case_2, ..., case_n} ∈ CB |
       (sim(query, case_i) ≥ simThreshold)}
3: if (| Res | ≥ numCases) then
4:    aSelectedCluster =
          chooseCluster(Res, Clusters, aClusterReuseRule)
5:    Res' = {case_i ∈ Res |
          (case_i.clusterLabel == aSelectedCluster)}
6:    if (| Res' | ≥ numCases) then
7:       aGameAction =
             reuseGameAction(Res', aReuseRule)
8:    end if
9: end if
10: return aGameAction
```

To compute reuse policies involving the selection of clusters, this work identified and represented in the cases' structure different combinations between game states and actions for each existing decision-making problem in Truco. In each state, game decisions are taken from analyzing different game information. To identify different game tactics used in these different contexts, 14 different sets of clusters were formed and used by the algorithms that implement the Truco agents. Using the cases stored in the case base as input, the number of clusters in each of the 14 problems was defined using the WCSS technique. The sum of the 14 decision-making Truco problems resulted in 67 different clusters. To compute the proposed reuse policies, an attribute representing the group each case belongs to was included in the case representation model. This multi-valued attribute represents the group the represented case belongs to in each game problem situation.

B. Cases and Clusters Reuse Policies

Various forms of combining the different reuse criteria for selecting groups of cases and game actions were investigated in this work. In doing so, we explored the two-step reuse model (Algorithm 1) to ensure that game actions occurring in different problem situations are not computed as similar game moves by the reuse policies. For example, a player may have cards that belong to either high/low-value card categories. Because of this, he/she plays the highest card among the ones in his/hers current hand. Playing the highest card can be understood as a similar game move in different game problem situations, although the hand configuration is relatively different. If the query and/or the retrieval algorithm didn't allow distinguishing between these distinct hand configurations, the reuse policy might consider that playing the highest card would be the best game action to be reused. However, the reuse policy has more information to identify that a different game action is a better option whenever the playing hands are organized into different clusters. As identified through the cluster analysis, different groups can capture cases with similar game-playing interactions and dissimilar from other cases organized in other groups.

The reuse criterion that computes the majority rule (MJ) was explored in this work as it delivered the best game-playing performance in [6]. The MJ criterion selects the cluster/game action recorded in the most similar retrieved cases. When applied directly to the game action selection, we called it the majority solution (MJS). When selecting the cluster of cases, we called it the majority rule with clusters (MJC). Given a cluster/solution vector $S = (s_1, s_2, \ldots, s_n)$, the MJ cluster/solution reuse policy selects the solution corresponding to $argmax\ S$.

A reuse criterion that explores probability computations is analyzed in [6], [17]. Our work considers the probability of each cluster/solution to be selected, as such probabilities are estimated with the retrieved cases for the given query. From this, a lottery function uses these estimates to compute the chances of each cluster/solution. We call this reuse criterion the probability lottery (PL). The PL criterion in the game action selection is called probability lottery solution (PLS). The selection of the case group is called probability lottery with clusters (PLC).

In our implementations, the number of each cluster/solution occurrence is computed and divided by the total number of clusters/solutions in the list of retrieved cases. Then, the results are taken as input into the lottery function. This function considers the chances of each cluster/solution being selected. In the end, the game action/cluster returned by the lottery computation is taken as an answer for the current query. Considering the probabilistic cluster/solution vector $P = (P(s_1), P(s_2), \ldots, P(s_n))$, the probability lottery (PL) cluster/solution reuse policy selects the cluster/solution is such that corresponds to $randomP$.

We also explore reuse policies to select game actions with higher chances of success. These policies rely on a) the game outcomes resulting from the use of a particular game action [5] and b) the probability estimates for the use of alternative solutions [6], [17]. The union of these techniques resulted in the reuse criterion called probability victory (PV). The number of wins each solution/cluster has achieved according to the retrieved cases for the given query is considered. This criterion returns the cluster/game action with the highest chance of success. We called it the probability victory solution (PVS) when selecting which solution should be reused. Like other policies, the probability victory with clusters (PVC) is used to select the case group to be analyzed. After computing the number of occurrences of each item (either cluster or game action) in the

retrieved cases, the number of wins to each is computed. The solution/cluster with the highest number of wins is selected. Given a cluster/solution vector $S = (s_1, s_2, \ldots, s_n)$ and the outcome vector $O = \{o_1, o_2, \ldots, o_n\}$, the probability victory (PV) reuse policy selects the cluster/solution s_i that corresponds to $argmax\ S$ considering $o_i == win$.

Game actions that involve making bets aim to earn points in Truco. In addition to having either victory or defeat consequences, these actions also result in points earned/lost. In [5], [6], the value lost/earned in each played game action is stored as an attribute in the case representation. Then, the reuse criterion analyzes the total sum of points for each cluster/solution represented in the retrieved cases for the query. This criterion is focused on the analysis of betting actions, limiting its applicability to the choice of other game solutions. Thus, it explores the probability victory (PV) criterion when dealing with game actions with such binary consequences. In this work, we called it the number of points (NP). It is the number of points solution (NPS) when selecting the game action to be reused and the number of points with clusters (NPC) when selecting the case group. This method computes the sum of the gains/losses resulting from each action/cluster. Then, the game action/cluster that resulted in the best outcomes is returned. Given a cluster/solution vector $S = (s_1, s_2, \ldots, s_n)$ and the outcome vector $O = \{o_1, o_2, \ldots, o_n\}$, the NP reuse policy selects the cluster/solution s_i that corresponds to $argmax\ S$ with the highest o_i.

TABLE I: Reuse models where different reuse policies are used to select clusters and solutions (i.e., game actions).

	Reuse policies	**Selecting the cluster**	**Selecting the solution**
(A)	MJS	-	MJ (majority rule)
	PLS	-	PL (probability lottery)
	PVS	-	PV (probability victory)
	NPS	-	NP (number of points)
(B)	MJCS	MJ (majority rule)	MJ (majority rule)
	PLCS	PL (probability lottery)	PL (probability lottery)
	PVCS	PV (probability victory)	PV (probability victory)
	NPCS	NP (number of points)	NP (number of points)
(C_1)	NPCMJS	NP (number of points)	MJ (majority rule)
	NPCPVS		PV(probability victory)
	NPCPLS		PL (probability lottery)
(C_2)	MJCNPS	MJ (majority rule)	NP (number of points)
	MJCPVS		PV(probability victory)
	MJCPLS		PL (probability lottery)
(C_3)	PLCNPS	PL (probability lottery)	NP (number of points)
	PLCMJS		MJ (majority rule)
	PLCPVS		PV (probability victory)
(C_4)	PVCNPS	PV (probability victory)	NP (number of points)
	PVCMJS		MJ (majority rule)
	PVCPLS		PL (probability lottery)

Each different reuse policy delivers different game-playing behaviors. However, knowing which policies can achieve the best results in a set of Truco matches is not yet possible. In Table I, combinations between reuse criteria for the selection of clusters and solutions were investigated in this work: (A) reuse policies for the direct selection of the past case solution, where the resulting agents just compute the standard policies; according to the two-step case reuse model detailed in Algorithm 1: (B) reuse policies where the same reuse criterion is computed twice and (C) novel reuse policies combining different reuse criteria to select clusters and case solutions.

IV. Experiments and Results

Experiments were developed in our project using the test methodology from the Annual Computer Poker Competition (ACPC) [4], [26]. There, the implemented agents are subjected to "duplicate game matches". It means that a given set of cards A is given to player J, and a set of cards B is given to player K. After the end of the first competition round, the match is repeated using the inverted sets of cards, where player J receives the set of cards B and player K receives the set of cards A. At the end of the second competition round, the cards received by each player are considered equivalent, allowing the use of the same set of cards to evaluate the agent's effectiveness (where the game-playing performance is assessed in terms of the number of wins). This strategy is well-tested in card-game tournaments, where Truco and Poker have card-playing similarities that warrant using it.

Initial tests focused on the solution reuse techniques with the EBCB, containing 3,195 cases collected with human players. Then, this case base was expanded to 27,515 cases to form the "*learned case base (LCB)*", where 1,833 self-play Truco matches were disputed between implemented agents. To form the LCB, the retrieval algorithm had to return at least 100 similar cases, and all 100 returned cases would need to have a similarity of at least 98% with the query. When these conditions were not satisfied, the dispute of the self-play-based Truco's hand was retained as new cases in the case base. In each match, the reuse policies used by each competitor were randomly selected, resulting in disputes between opponents with different game-playing characteristics.

When clustering-based reuse policies were computed to support the cases' collection, each generated case was assigned to the defined cluster. This assignment was made in each one of the clusters representing the 14 Truco decision-making problems. In the case model, a case attribute representing the cluster that the case belongs to was automatically filled out during the case retention. In particular, the value of this attribute was filled out with the identification of the group whose centroid was the most similar to the case to be stored there in the case base. The number and composition of case clusters were performed at the end of each played match in the self-play learning.

A. The selection of the promising agents

The 20 implemented agents disputed the duplicate matches, where all agents played against every other. The agents' performance according to the two-step reuse model was compared to the performance of the agents implemented using more standard techniques [6], [7]. The agents who won more than 50% of the played matches were selected to support further tests. This testing stage aimed to select the most promising set of agents using the EBCB initially collected with human

players. Then, they were used in the self-play learning to support the collection of new Truco cases and significantly expand the EBCB. The implemented agents were subjected to 25 duplicate matches with this case base, where all possible opponent configurations were executed. Each agent played 950 matches, with a total of 19,000 games.

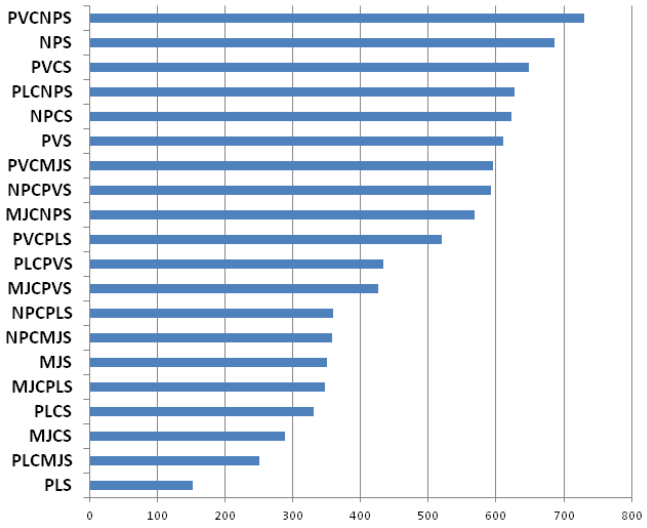

Fig. 2: Results from Truco matches disputed between all-against-all bots implementing different reuse policies (all bots used the EBCB).

Fig. 2 shows the accumulated performance results of the initially implemented 20 agents. Only 10 agents have won more than half of the matches, where these agents were selected to further tests. Among them, 8 agents used the two-step reuse model investigated in our work. As expected, 2 standard reuse techniques also had a relevant game-playing performance. In addition to the PVC criterion, the NPS criterion is present in the 4 winning reuse policies. In turn, the PVS criterion is represented in 3 selected policies, and the NPC criterion is in 2 selected policies. The PLC, PLS, MJC, and MJS criteria are represented in only 1 of the 10 selected reuse policies. The PVCNPS agent had 44 more wins than the NPS agent (second place). The PVCNPS agent used the probability victory criterion in selecting the cluster and the number of points criterion in selecting the game actions found in the retrieved cases belonging to that selected cluster. Thus, the game action that achieved the largest gain was selected, where the selection was based on the Truco hands recorded in the cases belonging to the cluster with the highest probability of victory. Ultimately, these tests also aimed to select agents with different game-playing behavior profiles.

B. The development of the self-play learning

The previously identified top 10 agents competed in duplicate Truco matches in this testing stage. Here, agents implemented with identical reuse policies disputed a set of matches. One of the competitors used the EBCB initially collected with human players. The other used the LCB resulting from the developed self-play learning. The aim is to analyze the game-playing effects resulting from self-play case learning.

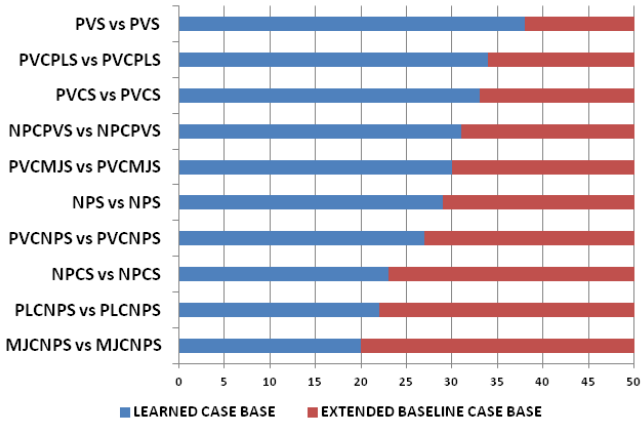

Fig. 3: Results from Truco matches disputed between the top 10 selected bots implementing the same reuse policies. While one bot used the LCB, its opponent used the EBCB.

Fig. 3 presents the performance of each of the reuse policies implemented by the 10 agents along with their different case bases. It shows that the LCB permitted agents (competitors with identical reuse policies but different case bases) to improve their game-playing skills. Among the 500 disputed matches, 287 were won by the agents using the LCB.

The performance increase of the PVS policy may be related to the increase in the number of cases retrieved from the case base due to the query. This higher number of retrieved cases allowed better probability estimates to be computed from past game decisions. This also occurred in the PVCNPS, PVCMJS, and PVCPLS policies, where the game action selection is executed according to the selected case group with improved decision security due to the PVC criterion.

The PVCNPS agent game-playing performance also improved due to the LCB. This agent makes decisions based on higher gains according to the group of cases with a higher probability of victory. This also occurred in the PVCMJS and PVCPLS agents because a) the PVCMJS agent selects the most commonly executed game action (MJS) and b) the PVCPLS agent explores the lottery function, which is based on the chances of each game action (PLS). Moreover, these criteria are all computed according to the retrieved cases that belong to the group with the highest probability of victory.

For the NPCPVS policy, the selection of the game action with the highest chance of victory within the group that provided the highest gains also benefited from expanding the case base. Even when risky game actions are within the group with the highest gains, the PV criterion only suggests game actions with a greater probability of victory than defeat.

Finally, the NPS policy's use of the LCB favored the reuse of game actions by analyzing the larger number of retrieved cases showing occurrences of defeat. The fact that the case base stores more defeat cases caused the NPS policy to recommend game actions typical of less offensive game-playing

moves. In summary, among the 10 tested reuse policies, 7 had a positive game-playing evolution using the LCB.

C. The identification of the best reuse policies

The same top 10 agents used in previous tests were subjected to new Truco matches. With all-against-all disputes and only the use of the LCB, these tests evaluate these agents' performance. Such selected agents disputed matches where all possible opponent settings were set up. Each agent played 450 matches of 2,250 developed.

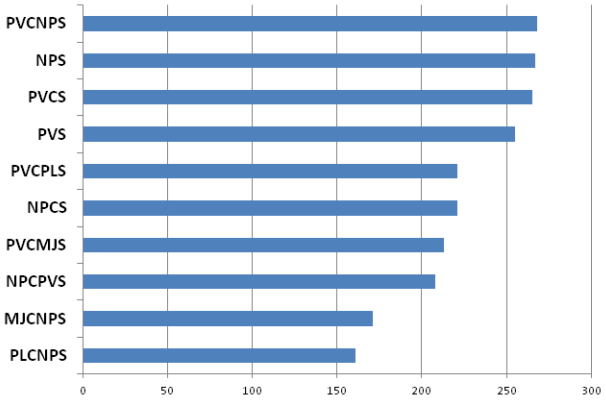

Fig. 4: Results from Truco matches disputed between the top 10 selected bots implementing the same reuse policies (all bots used the LCB).

Fig. 4 shows that the PVCNPS, NPS, PVCS, and PVS agents won more than half of the played matches. Among them, the PVCNPS and PVCS agents were based on the two-step reuse model, and the NPS and PVS agents were based on the standard reuse model.

Fig. 5a shows that the PVCNPS agent obtained the highest score difference in wins, the lowest in losses, and the highest number of wins. In general, the combination of a decision-making criterion considered conservative for selecting the cluster and an aggressive one for selecting the game action allowed the PVCNPS agent to defeat most opponents. The positive effect of the PVCNPS policy was even greater when the opposing agents relied on conservative game-playing behaviors, i.e., the opponents who tended to bet only when they had strong playing hands or when the opponents had used aggressive game-playing policies that relied on the analysis of case clusters in which very risky game moves were stored.

Fig. 5b shows that the NPS agent obtained the second-best result in these tests. By directly applying the NP criterion to the solution selection, this policy allowed the agent to behave more aggressively than when the PVCNPS criterion was used. While in specific scenarios, the NPS agent allowed one to achieve better results, in others, the NPS agent was penalized because the NPS policy tended to return risky game moves, which could not win the dispute. The same actions that provided the most significant gains were the ones that caused the greatest losses.

Fig. 5c shows that the PVCS agent obtained the third-highest number of wins. The PVCS criterion enabled the implementation of an agent capable of prioritizing game actions with a high probability of victory. Despite implementing a policy that could be considered conservative, the PVCS agent obtained the third-largest difference in the winning scores.

Fig. 5d shows that the PVS agent obtained the lowest number of victories. However, this agent obtained the third-lowest difference in the defeat scores. In summary, combining the PVC and NPS criteria allowed for maximizing gains and minimizing losses in the disputed matches.

V. Concluding Remarks

The stochastic nature and partial vision of the card games, extensively observed in Truco, present a challenging scenario for investigating cases and clusters and the proposal of novel reuse policies for game-playing algorithms. As described in this work, one reuse criterion is applied to select the cluster capturing the game-playing behaviors, while another uses the cases belonging to the selected cluster to determine the game action to be executed by the agent. This work also details the self-play approach for retaining reusable game-playing experiences in larger case bases. The main experimental results show the game-playing behavior resulting from the probability victory (PV) criterion to select a cluster of cases and the number of points (NP) criterion to select a game action was the most promising. With the help of statistical methods, we plan to evaluate our agents in future Truco competitions against agents implemented with other machine learning techniques. We will also expand the contributions of this work with the investigation of multiplayer Truco games.

References

[1] M. M. Richter and R. O. Weber, *Case-Based Reasoning: A Textbook*. Springer, 2013.

[2] A. E. Ezugwu, A. M. Ikotun, O. O. Oyelade, L. Abualigah, J. O. Agushaka, C. I. Eke, and A. A. Akinyelu, "A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects," *Engineering Applications of Artificial Intelligence*, vol. 110, p. 104743, 2022.

[3] L. L. Winne, *Truco*. Ciudad Autónoma de Buenos Aires: Ediciones Godot, 2017, bilingual edition.

[4] J. Rubin and I. Watson, "Case-based strategies in computer poker," *AI Communications*, vol. 25, no. 1, pp. 19–48, 2012.

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

[6] J. Rubin and I. Watson, "Similarity-based retrieval and solution reuse policies in the game of texas hold'em," in *Case-Based Reasoning. Research and Development: 18th Int. Conference on Case-Based Reasoning, ICCBR 2010*. Springer, 2010, pp. 465–479.

[7] G. B. Paulus, J. V. C. 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)*. IEEE, 2019, pp. 1361–1365.

[8] R. C. B. Moral, G. B. Paulus, J. V. C. Assunção, and L. A. L. Silva, "Investigating case learning techniques for agents to play the card game of truco," in *2020 19th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, 2020, pp. 107–116.

[9] D. P. Vargas, G. B. Paulus, and L. A. L. Silva, "Active learning and case-based reasoning for the deceptive play in the card game of truco," in *Intelligent Systems*, A. Britto and K. Valdivia Delgado, Eds. Cham: Springer International Publishing, 2021, pp. 313–327.

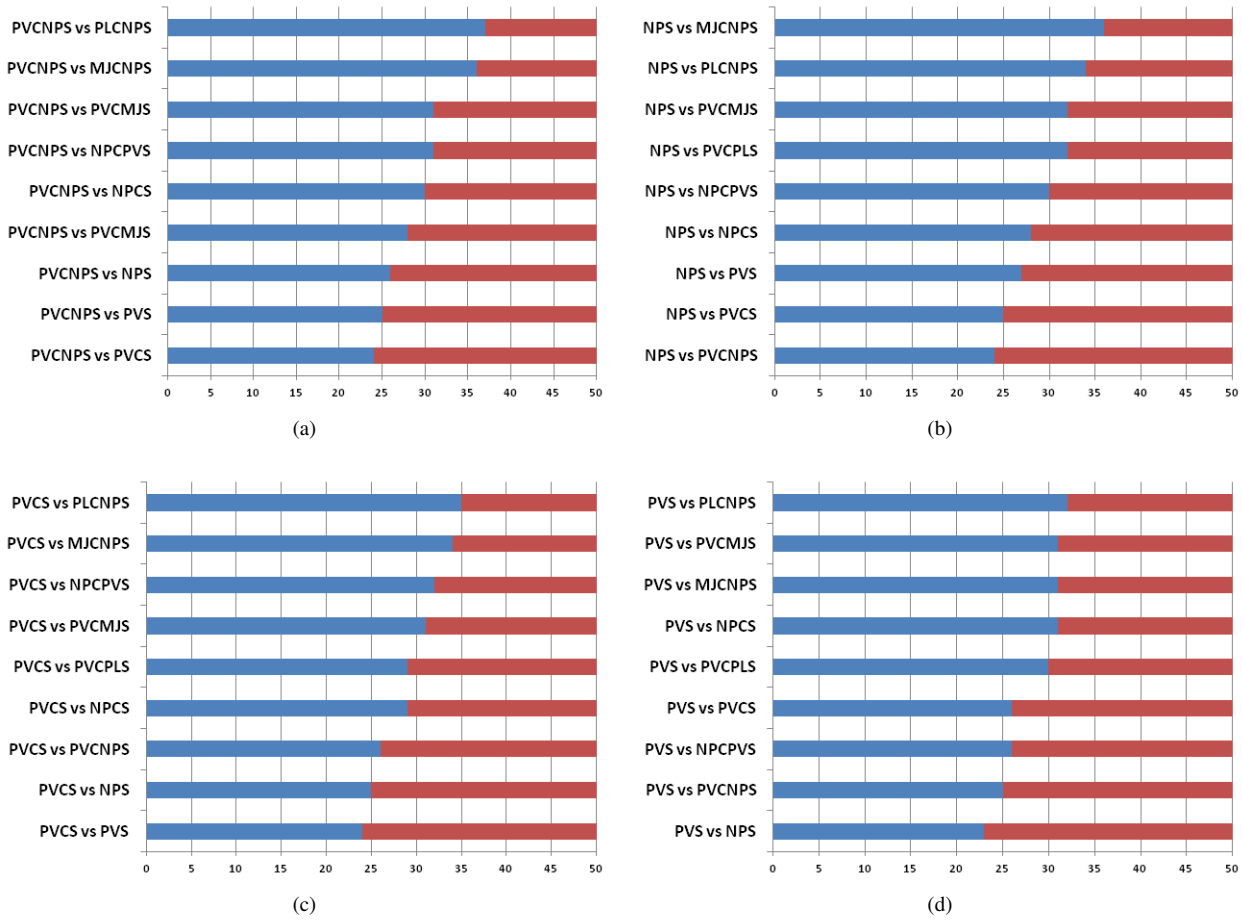

Fig. 5: Performance of the top 4 implemented reuse policies. Results from the all-against-all Truco matches disputed between the top-10 selected BOTs implementing different reuse policies (all BOTs used the LCB).

[10] 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," 2019. [Online]. Available: https://arxiv.org/abs/1906.04439

[11] S. Ontañón and M. W. Floyd, "A comparison of case acquisition strategies for learning from observations of state-based experts," in *Proc. of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2013, St. Pete Beach, Florida, USA, 2013*, C. Boonthum-Denecke and G. M. Youngblood, Eds. AAAI Press, 2013.

[12] 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," in *2017 12th International Conference on Computer Engineering and Systems (ICCES)*, 2017, pp. 337–346.

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

[14] J. H. Powell, B. M. Hauff, and J. D. Hastings, "Evaluating the effectiveness of exploration and accumulated experience in automatic case elicitation," in *Case-Based Reasoning Research and Development*, H. Muñoz-Ávila and F. Ricci, Eds. Springer Berlin Heidelberg, 2005.

[15] J. Rubin and I. Watson, "Computer poker: A review," *Artificial Intelligence*, vol. 175, no. 5, pp. 958–987, 2011.

[16] V. Ventos, Y. Costel, O. Teytaud, and S. Thépaut Ventos, "Boosting a bridge artificial intelligence," in *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2017, pp. 1280–1287.

[17] J. Rubin and I. D. Watson, "Investigating the effectiveness of applying case-based reasoning to the game of texas hold'em." in *FLAIRS Conference*, 2007, pp. 417–422.

[18] B. Smyth and E. McKenna, "Competence models and the maintenance problem," *Computational Intelligence*, vol. 17, no. 2, pp. 235–249, 2001.

[19] N. Brown and T. Sandholm, "Superhuman ai for multiplayer poker," *Science*, vol. 365, no. 6456, pp. 885–890, 2019.

[20] G. Khussainova, S. Petrovic, and R. Jagannathan, "Retrieval with clustering in a case-based reasoning system for radiotherapy treatment planning," in *Journal of Physics: Conference Series*, vol. 616, no. 1. IOP Publishing, 2015.

[21] J. M. Juarez, S. Craw, J. R. Lopez-Delgado, and M. Campos, "Maintenance of case bases: Current algorithms after fifty years," in *Proc. of the Twenty-Seventh Int. Joint Conference on Artificial Intelligence, IJCAI-18*. Int. Joint Conferences on Artificial Intelligence Organization, 2018, pp. 5457–5463.

[22] C. Bauckhage, A. Drachen, and R. Sifa, "Clustering game behavior data," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 7, no. 3, pp. 266–278, 2015.

[23] D. Wehr and J. Denzinger, "Mining game logs to create a playbook for unit ais," *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 391–398, 2015.

[24] L. F. Teófilo and L. P. Reis, "Identifying players strategies in no limit texas holdém poker through the analysis of individual moves," 2013. [Online]. Available: https://arxiv.org/abs/1301.5943

[25] M. W. Floyd and B. Esfandiari, "A case-based reasoning framework for developing agents using learning by observation," in *IEEE 23rd Int. Conference on Tools with Artificial Intelligence*, 2011, pp. 531–538.

[26] N. Bard, J. Hawkin, J. Rubin, and M. Zinkevich, "The annual computer poker competition," *AI Magazine*, vol. 34, no. 2, pp. 112–112, 2013.