Similarity-Based Retrieval and Solution Re-use Policies in the Game of Texas Hold'em*

Jonathan Rubin and Ian Watson

Department of Computer Science
University of Auckland, New Zealand
jrub001@aucklanduni.ac.nz, ian@cs.auckland.ac.nz
http://www.cs.auckland.ac.nz/research/gameai

Abstract. In previous papers we have presented our autonomous poker playing agent (SARTRE) that uses a *memory-based* approach to create a betting strategy for two-player, limit Texas Hold'em. SARTRE participated in the 2009 IJCAI Computer Poker Competition where the system was thoroughly evaluated by challenging a range of other computerised opponents. Since the competition SARTRE has undergone case-based maintenance. In this paper we present results from the 2009 Computer Poker Competition and describe the latest modifications and improvements to the system. Specifically, we investigate two claims: the first that *modifying the solution representation results in changes to the problem coverage* and the second that *different policies for re-using solutions leads to changes in performance*. Three separate *solution re-use policies* for making betting decisions are introduced and evaluated. We conclude by presenting results of self-play experiments between the *pre* and *post* maintenance systems.

1 Introduction

Games offer a well suited domain for Artificial Intelligence (AI) investigation and experimentation [1] due to the fact that a game is usually composed of several well-defined rules which players must adhere to. Most games have precise goals and objectives which players must meet to succeed. For a large majority of games the rules imposed are quite simple, yet the game play itself involves a large number of very complex strategies. Furthermore, a performance metric is naturally embedded into the game itself. Success can therefore easily be measured by factors such as the amount of games won or the ability to beat certain opponents.

Games are often classified by the amount of information available to the players. If a player has access to all the information they require about the game during play then the game can be classified as having *perfect information*. However, if some of that information is hidden from the player the game is known

* If you wish to challenge the latest version of SARTRE, you may do so at http://www.cs.auckland.ac.nz/poker/

as having *imperfect information*. Take for example the game of chess. Chess is a game of *perfect information* because each player can look down upon the board and obtain all the information necessary to make their playing decisions. On the other hand, the game of poker is a game of *imperfect information*. In poker players are given cards which only they can see, therefore players now have to make decisions based on hidden information because they cannot see their opponents' cards.

Games can be further classified as either *deterministic* or *stochastic*. If a game contains chance elements, such as the roll of a dice, this introduces randomness into the game. These types of games are known as *stochastic* games and examples include bridge, backgammon and poker. The absence of these chance elements ensures the game is *deterministic*. Games such as chess, checkers and go are examples of deterministic games.

Poker is a *sequential*, *stochastic* game with *imperfect information*. It is *sequential* because players choose their actions in sequence. It is *stochastic* because the shuffling of cards introduces randomness into the game. It is a game of *imperfect information* because players cannot see their opponent's cards, therefore players need to make decisions based on hidden information. Given the relatively simple rules of the game there is an enormous amount of subtle and sophisticated scenarios that can occur during a hand of play (this is particularly true of the Texas Hold'em variation).

In previous papers [2, 3] we introduced our autonomous poker agent SARTRE (Similarity Assessment Reasoning for Texas hold'em via Recall of Experience) that plays the game of 2-player, limit Texas Hold'em. SARTRE constructs a poker betting strategy using a *memory-based* approach where cases are stored in a database which describe past game state information, along with the betting decisions made and the eventual outcome of those decisions. SARTRE chooses a betting action by consulting this memory of past games and retrieving similar game states. Once the most similar game state has been found its solution is re-used.

In 2009 we submitted SARTRE to the IJCAI Computer Poker Competition, where the system was thoroughly evaluated by challenging many different types of opponents. In this paper we present the results of the 2009 limit competitions in which SARTRE participated. Since the competition, case-base maintenance [4] has been conducted on the SARTRE system. This maintenance phase resulted in changes to SARTRE's solution representation. In this paper we introduce the latest modifications and improvements to the system. In particular, we address the following two claims:

Claim 1: Changes in solution representation results in changes to problem coverage.

Claim 2: Different policies for re-using solutions leads to changes in performance.

We introduce 3 separate policies for making betting decisions, which we label *solution re-use policies*. The performance of each *solution re-use policy* is evalu-

ated and the experimental results presented, along with the results of self-play experiments between the *pre-maintenance* and *post-maintenance* systems.

The paper proceeds as follows. Section 2 briefly describes the rules of Texas Hold'em. Section 3 reviews related work and introduces two broad types of poker strategies. An overview of our *memory-based* approach is given in Section 4, along with details regarding the maintenance that the system has undergone since the 2009 Computer Poker Competition. For the purposes of this paper we will refer to the version of SARTRE that participated in the 2009 competition as SARTRE-1 and the updated, post-maintenance system as SARTRE-2. Section 5 presents the results of the 2009 Computer Poker Competition along with further self-play experiments between SARTRE-1 and SARTRE-2. Conclusions of the work are discussed in Section 7 and avenues for future research discussed in Section 8.

2 Texas Hold'em

Here we briefly describe the game of Texas Hold'em, highlighting some of the common terms which are used throughout this work. For a more in depth introduction to Texas Hold'em consult [5].

The game of Texas Hold'em is played in 4 stages – **preflop**, **flop**, **turn** and **river**. During the preflop all players at the table are dealt two **hole cards**, which only they can see. Before any betting takes place, two forced bets are contributed to the pot, i.e. the **small blind** and the **big blind**. The big blind is typically double that of the small blind. The player to the left of the big blind, known as **under the gun**, then begins the betting by either folding, calling or raising. The possible betting actions common to all variations of poker are described as follows:

Fold: When a player contributes no further chips to the pot and abandons their hand and any right to contest the chips which have been added to the pot.

Check/Call: When a player commits the minimum amount of chips possible in order to stay in the hand and continue to contest the pot. A check requires a commitment of zero further chips, whereas a call requires an amount greater than zero.

Bet/Raise: When a player commits greater than the minimum amount of chips necessary to stay in the hand. When the player could have checked, but decides to invest further chips in the pot, this is known as a bet. When the player could have called a bet, but decides to invest further chips in the pot, this is known as a raise.

In a **limit** game all bets are in increments of a certain amount. In a **no limit** game, players can wager up to the total amount of chips they possess in front of them. Once the betting is complete, as long as at least two players still remain in the hand, play continues onto the next stage. Each further stage involves the drawing of **community cards** from the deck. Players combine their hole cards with the public community cards to form their best 5 card poker hand.

The number of community cards revealed at each stage is as follows: **flop** – 3 community cards, **turn** – 1 community card, **river** – 1 community card. Each stage also involves its own round of betting and as long as there are players left who have not folded their hands, play continues. A **showdown** occurs after the river where the remaining players reveal their hole cards and the player with the best hand wins all the chips in the pot. If two or more players have the same best hand then the pot is split amongst the winners. The list of poker hand categories in ascending order of strength is as follows: *high-card*, *one-pair*, *two-pair*, *three-of-a-kind*, *straight*, *flush*, *full-house*, *four-of-a-kind*, *straight-flush*.

3 Related Work

There are two main types of strategies that a poker agent may employ:

1. A **Nash equilibrium** strategy, or
2. an **exploitive** strategy

A **strategy** in this context refers to a mapping between game states and the actions that an agent will take at that game state. Typically, an agent's strategy consists of specifying a **probability triple** at every game state. A probability triple, *(f,c,r)*, specifies the proportion of the time an agent will either fold (f), check/call (c) or bet/raise (r) at a particular point in the game.

A **Nash equilibrium** is a robust, static strategy that attempts to limit its exploitability against a worst-case opponent. In general, a set of strategies are said to be in *equilibrium* if the result of one player diverging from their equilibrium strategy (while all other players stick to their current strategy) results in a negative impact on the expected value for the player who modified their strategy [6]. Currently, it is intractable to compute exact Nash equilibria for full-scale Texas Hold'em [7], but by applying simplifying abstractions to the game it is possible to approximate a Nash equilibrium strategy. The University of Alberta Computer Poker Research Group[1] (CPRG) were the first to approximate a Nash equilibrium for the full-scale game of two-player Texas Hold'em [8]. One of the outcomes of this research was the poker agent Sparbot, which is available within the commercial software product **Poker Academy Pro 2.5**[2]. GS1 [9] and GS2 [10] developed by Andrew Gilpin and Thomas Sandholm at Carnegie Mellon University are also examples of early attempts to derive approximate equilibrium solutions for limit hold'em.

Over the years it has become possible to construct and solve larger mathematical models of the poker game tree via improved iterative procedures, such as counterfactual regret minimization (CFRM) [11, 7]. Typically, by applying fewer simplifying abstractions to the game model, stronger Nash equilibrium strategies have been produced. The University of Alberta CPRG's latest Nash based agent is Hyperborean-Eqm and it was constructed via the CFRM algorithm. The

[1] http://poker.cs.ualberta.ca/
[2] http://www.poker-academy.com/

winner of the limit equilibrium division at the 2009 IJCAI Computer Poker Competition was GGValuta. GGValuta was developed by a team of students from the University of Bucharest based on the CFRM algorithm [12].

As a Nash equilibrium strategy assumes an unknown, worst-case opponent, it will limit its own exploitability at the expense of taking advantage of weaker opponents. Hence, while this sort of strategy may not lose, it will also not win by as much as it could against weaker opponents. On the other hand, a player that attempts to isolate the weaknesses of their opponent and capitalize on those weaknesses is said to employ an **exploitive** (or **maximal**) strategy. This is typically achieved by constructing a model of an opponent and using it to inform future actions. A consequence of an exploitive strategy is that it no longer plays near the equilibrium and hence is vulnerable to exploitation itself, especially if the model of the opponent is incorrect or no longer valid. Vexbot [13] is an example of an exploitive poker agent that has been created using opponent modeling and imperfect information game tree search. Vexbot is also available within Poker Academy Pro 2.5.

Other approaches used to construct exploitive agents include the use of Monte-Carlo simulation [14, 15] and the Monte-Carlo Tree Search algorithm [16], which involve the evaluation of nodes in the game tree by drawing repeated random samples. More recent exploitive strategies such as Restricted Nash Response (RNR) [17, 7] and Data Biased Response (DBR) [18] attempt to optimise the trade-off between the robust nature of Nash equilibrium strategies and the power of exploitive strategies that rely on opponent modelling. Finally, various machine learning approaches have been used to construct strong poker agents. For example, the winner of the limit bankroll division of the 2009 IJCAI Computer Poker Competition was MANZANA, developed by hobbyist Marv Andersen. MANZANA employed the use of artificial neural networks trained on data from the best agent of the previous year's competition [12].

3.1 CBR Motivation

The focus of this paper is on generating case-based poker strategies. When a new problem is encountered similar cases are retrieved from the case-base of poker hands and their solutions are adapted or re-used to solve the problem. As Nash equilibrium-based strategies are very large a current goal of our research is to determine how close this type of strategy can be approximated with a compact case-base that relies on finding similar cases and generalising the observed actions. Case-based strategies can easily be used to approximate the play of a selected "expert" or group of "experts" via observation and case generation. Expert players can be human players or other computerised agents. By simply updating the case-base, different types of players can be modelled without relying on the creation of complicated mathematical models or algorithms.

CASPER [19, 20], is an example of a previous poker agent constructed using a case-based strategy. CASPER was designed to play 10-player limit hold'em. Our latest poker agent, SARTRE [2] has been specialised to play 2-player, limit

hold'em. We refer to the approach used by SARTRE as a *memory-based* approach and it is introduced in the next section.

4 Our Approach

4.1 Overview

An overview of the *memory-based* approach used by SARTRE is as follows:

- A database of cases is built by observing actual poker hands. Each case consists of a collection of attribute-value pairs that encapsulate game state information.
- Separate case-bases are constructed for each round of play (*preflop*, *flop*, *turn*, *river*).
- When SARTRE is required to make a betting decision a **target case** is created to describe the current state of the game and the appropriate case-base (collection of **source cases**) is searched to locate similar cases.
- A betting decision is made by employing 1 of 3 solution re-use policies: *probabilistic*, *majority-rules* or *best-outcome* (refer to Section 4.5).

The details of the above approach are now presented.

4.2 Case Representation

Table 1 depicts the post-flop case representation[3]. Every case is made up of 3 attributes that capture information about the current game state. The attributes were selected by the authors as they are believed to encapsulate the salient aspects of the game in a concise manner. The first attribute (**hand type**) classifies the 5-card hand of a player into a category which represents information about its current strength and ability to improve (such as whether the opponent has a flush or straight draw). The next attribute (**betting sequence**) simply enumerates the betting that has been witnessed up till the current point in the hand. An r stands for a bet or a raise, a c stands for a check or a call and a hyphen delimits the betting rounds. The final attribute (**board texture**) highlights important information about the public community cards such as whether there are 3 or more cards of the same suit present on the board and hence a player could have a flush (i.e. ***Flush-Possible*** and ***Flush-Highly-Possible***)[4].

Each case also contains a solution, which is made up of an **action** triple and an **outcome** triple. The **action** triple suggests a probability distribution, *(f, c, r)*, over betting actions that the agent should select given the current state of the game. The **outcome** triple records the average quantitative profit or loss that has been observed in the past given the various betting decisions. Outcomes that have never been observed are labelled with $-\infty$.

[3] Pre-flop cases are slightly different in that they only contain **hand type** and **betting sequence** features, where **hand type** refers to 1 of the standard 169 pre-flop equivalence classes.

[4] A complete listing of all the **hand type** and **board texture** categories can be found at www.cs.auckland.ac.nz/research/gameai/sartreinfo.html

Table 1. A case is made up of three attribute-value pairs which describe the current state of the game. A solution consists of an action triple and an outcome triple, which records the average numerical value of applying the action in this case (-∞ refers to an unknown outcome).

Attribute	Type	Example
1. Hand Type	Class	*High-Card*, *Pair*, *Two-Pair*, *Set*, *Flush*, *Pair-with-Flush-Draw*, *High-Card-with-Straight-Draw*, ...
2. Betting Sequence	String	*rc-c*, *crrc-crrc-cc-*, *r*, ...
3. Board Texture	Class	*No-Salient*, *Flush-Possible*, *Straight-Possible*, *Flush-Highly-Possible*, ...
Action	Triple	(0.0, 0.5, 0.5), (1.0, 0.0, 0.0), ...
Outcome	Triple	(-∞, 4.3 , 15.6), (-2.0,-∞,-∞), ...

Claim 1 in section 1 states that: *changes in solution representation results in changes to problem coverage*. In particular, SARTRE-1 represented actions and outcomes as single characters or numerical values, respectively. At decision time all similar cases were retrieved and the solutions combined to form a probability triple. Using this representation SARTRE-1 was forced to store many duplicate cases within each of its case-bases. SARTRE-2 now uses full probability triples as its solution representation. To derive the probability triples, SARTRE-2 must first pre-process the training data. By pre-processing the training data SARTRE-2 allows a much more compact case-base size, due to the fact that it no longer retains many duplicate cases (see Table 2).

4.3 Similarity-Based Retrieval

The k-nearest neighbour algorithm (k-NN) is used to retrieve the most similar case from the case-base. The k-NN algorithm involves positioning the target case in an n-dimensional search space of source cases. Similarity between the target and source cases individual attributes is calculated using specific similarity metrics, described in detail below. The target case is compared to every case in the appropriate case-base and similarity computed for each attribute in the case.

For SARTRE-1 the value of k could vary between 0 to N, where N was only bounded by the number of cases in the case-base. If 0 cases were retrieved a default policy of Always-Call was adopted. Furthermore, SARTRE-1 required exact matches, otherwise the default policy was used. Given SARTRE-2's updated representation, k is now set to 1. SARTRE-2 no longer requires a default policy as the solution of the most similar case is always used, no matter what the similarity.

4.4 Similarity Metrics

In order to compute global similarity, each of the attributes represented in Table 1 requires their own local similarity metric. The attributes' specific local similarity metrics are described below, where the values within the brackets indicate the allowable similarity values:

Hand Type [0, 1]: Either a combination of cards are mapped into the same category as another combination of cards, in which case a similarity value of 1.0 is assigned, or they map into a separate category, in which case a similarity value of 0 is given. This same metric is used for both SARTRE-1 and SARTRE-2.

Board Texture [0, 1]: As above.

Betting Sequence [0, 0.8, 0.9, 1]: SARTRE-1 used a simplistic all or nothing similarity metric for the *betting sequence* attribute, where similarity was either 0 or 1. SARTRE-2 improves upon this simplistic metric by assigning stepped levels of similarity. The first level of similarity (level0) refers to the situation when one betting sequence exactly matches that of another. If the sequences do not exactly match the next level of similarity (level1) is evaluated. If two distinct betting sequences exactly match for the active betting round and for all previous betting rounds the total number of bets/raises made by each player are equal then level1 similarity is satisfied and a value of 0.9 is assigned. Consider the following example where the active betting round is the *turn* and the two betting sequences are:

1. *crrc-crrrrc-cr*
2. *rrc-rrrrc-cr*

Here, level0 is clearly incorrect as the sequences do not match exactly. However, for the active betting round (*cr*) the sequences do match. Furthermore, during the preflop (1. *crrc* and 2. *rrc*) both players made 1 raise each, albeit in a different order. During the flop (1. *crrrrc* and 2. *rrrrc*) both players now make 4 raises each. Given that the number of bets/raises in the previous rounds (preflop and flop) match, these two betting sequences would be assigned a similarity value of 0.9.

If level1 similarity was not satisfied the next level (level2) would be evaluated. Level2 similarity is less strict than level1 similarity as the previous betting rounds are no longer differentiated. Consider the river betting sequences:

1. *rrc-cc-cc-rrr*
2. *cc-rc-crc-rrr*

Once again the sequences for the active round (*rrr*) matches exactly. This time, the number of bets/raises in the preflop round are not equal (the same applies for the flop and the turn). Therefore, level1 similarity is not satisfied. However, the number of raises encountered for all the previous betting rounds combined (1. *rrc-cc-cc* and 2. *cc-rc-crc*) are the same for each player, namely 1 raise by each player. Hence, level2 similarity is satisfied and a similarity value of 0.8 would be assigned. Finally, if level0, level1 and level2 are not satisfied level3 is reached where a similarity value of 0 is assigned.

4.5 Solution Re-Use Policies

Once the above similarity metrics have been applied and a similar case retrieved, a betting decision is required. Claim 2 of this paper states: *different policies for re-using solutions leads to changes in performance.* To investigate this claim we have experimented with three separate approaches for making betting decisions, which we refer to as **solution re-use policies**. They are as follows:

Probabilistic: this solution re-use policy probabilistically selects a betting decision based on the *(f, c, r)* proportions specified by the solution of the retrieved case or cases. Betting decisions that have greater proportion values will be made more often then those with lower values.

Majority-Rules: the majority-rules solution re-use policy will re-use the decision that was made the majority of the time, as specified by the proportions contained within the solution of the most similar case or cases.

Best-Outcome: rather than re-use a case's action triple, the best-outcome solution re-use policy will make a betting decision based on the values in the outcome triple of the most similar case or cases. The betting decision which has the greatest average outcome overall is the one that is chosen.

4.6 Case-Base Construction

A beneficial property of the approach described above is the ability to easily *"plug-in"* different case-bases that have been trained on the hand history logs of various *"expert"* players. Typically, data obtained from a single Nash equilibrium-based player is chosen to train the system. A single expert is chosen, rather than multiple experts to avoid conflicting decisions. The expert's original strategy is then approximated by constructing cases and generalising the observed actions by retrieving the most similar case at decision time. The case-base used by SARTRE-1 was trained on the hand history logs of Hyperborean-Eqm, who was the winner of the limit equilibrium division at the 2008 Computer Poker Competition [21]. SARTRE-2 was trained on the hand history logs of MANZANA, which was the winner of the limit hold'em bankroll competition at the 2009 Computer Poker Competition [22]. Table 2 depicts the number of cases recorded for each separate betting round for both SARTRE-1 and SARTRE-2.

Given SARTRE-1's requirement to store a large number of cases, the amount of data used to train the system had to be restricted due to the corresponding storage costs. On the other hand, the use of probability triples along with preprocessing the training data ensures SARTRE-2 stores a more compact case-base. The reduced storage costs associated with SARTRE-2 allows a larger set of data to be used to train the system. The final result of this increased training is that SARTRE-2 encounters (and stores cases for) a wider range of problems than SARTRE-1.

Table 2. Total cases recorded per round for SARTRE-1 and SARTRE-2.

Round	Total Cases (SARTRE-1)	Total Cases (SARTRE-2)
Preflop	201,335	857
Flop	300,577	6,743
Turn	281,529	35,464
River	216,597	52,088
Total	1,000,038	95,152

5 Experimental Results

5.1 2009 IJCAI Computer Poker Competition

The Annual Computer Poker Competition[5] (CPC) has been held each year either at the AAAI or IJCAI conference since 2006. The CPC involves separate competitions for different varieties of Texas Hold'em, such as both limit and no-limit competitions as well as heads-up and multiple-opponent competitions. Entrance into the competition is open to anyone and the agents submitted typically represent the current state of the art in computer poker. The CPC uses a duplicate match structure. For a heads-up match a duplicate match proceeds as follows: N hands are played between two agents after which the agent's memories are wiped and the N hands played again, but in the reverse direction, i.e. the cards that were initially given to player A are instead given to player B and vice-versa. This way both players get to play both sets of N cards and this minimises the variance that is involved in simply playing a set of N hands in one direction only. In the 2009 competition, $N = 3000$ was used and many duplicate matches were played in order to achieve significant results.

The annual CPC typically involves two winner determination methods. The first is known as the **equilibrium competition** and the second is the **bankroll competition**. The equilibrium competition analyses the results of a set of matches in a way that rewards agents that play close to a Nash equilibrium. The bankroll competition rewards agents that are able to maximally exploit other agents, resulting in increased bankrolls.

Measurements are made in **small bets per hand** (sb/h), where the total number of small bets won or lost are divided by the total hands played. For example, assuming a \$10/\$20 hold'em game, imagine after 3000 hands a player has made a total profit of \$1800. To calculate the sb/h value, first the total profit is divided by the small bet (\$10) which is then divided by the total hands (3000) which gives a final value of +0.06 sb/h.

Final results for the 2009 IJCAI Computer Poker Competition are depicted in Table 3 for the 2-player limit hold'em bankroll and equilibrium divisions, respectively. For this competition SARTRE-1 used a majority-rules solution reuse policy and a default always-call policy when no similar cases were retrieved.

[5] http://www.computerpokercompetition.org/

Table 3. 2009 limit heads up bankroll and equilibrium results, respectively.

Place	Competitor	sb/h
1	MANZANA	0.186 ± 0.002
2	Hyperborean-BR	0.116 ± 0.002
3	GGValuta	0.110 ± 0.002
4	Hyperborean-Eqm	0.116 ± 0.002
5	Rockhopper	0.103 ± 0.002
6	*Sartre*	0.097 ± 0.002
7	Slumbot	0.096 ± 0.002
8	GS5	0.082 ± 0.002
9	AoBot	-0.002 ± 0.003
10	dcurbHU	-0.07 ± 0.002
11	LIDIA	-0.094 ± 0.002
12	GS5Dynamic	-0.201 ± 0.002

Place	Competitor
1	GGValuta
2	Hyperborean-Eqm
3	MANZANA
4	Rockhopper
5	Hyperborean-BR
6	Slumbot
7	*Sartre*
8	GS5
9	AoBot
10	GS5Dynamic
11	LIDIA
12	dcurbHU
13	Tommybot

Table 3 (left) shows that SARTRE-1 placed 6th out of 12 competitors in the bankroll division, achieving an average profit of 0.097 sb/h. Table 3 (right) depicts the final placings for the equilibrium competition, which rewards agents that play closer to a Nash equilibrium. In this division, SARTRE-1 placed 7th out of 13 competitors.

5.2 Self-play Experiments

This section presents the results of self-play experiments between:

- The 3 solution re-use policies of SARTRE-2, and
- The *pre* and *post* maintenance systems, SARTRE-1 and SARTRE-2.

The purpose of these experiments is to determine the validity of the claims put forward in section 1 and to investigate whether the maintenance process had any affect on the actual performance of the system.

Solution Re-Use Policies Table 4 presents the first set of results where the 3 solution re-use policies (described in Section 4.5) were played against each other. A round robin tournament structure was used, where each policy challenged every other policy. The figures presented are in small/bets per hand. Each match consisted of 3 separate duplicate matches, where $N = 3000$. Hence, in total 18,000 hands of poker were played between each competitor. All results are statistically significant.

Table 4 shows that the majority-rules policy outperforms its probabilistic and best-outcome counterparts. Of the three, best-outcome fairs the worst, losing all of its matches.

Table 4. Results of self play experiments between 3 solution re-use policies of SARTRE-2.

	Majority-rules	Probabilistic	Best-outcome	**Average**
Majority-rules		0.011 ± 0.005	0.076 ± 0.008	$\mathbf{0.044 \pm 0.006}$
Probabilistic	-0.011 ± 0.005		0.036 ± 0.009	$\mathbf{0.012 \pm 0.004}$
Best-outcome	-0.076 ± 0.008	-0.036 ± 0.009		$\mathbf{-0.056 \pm 0.005}$

SARTRE-1 *Vs.* SARTRE-2 Given the above results we selected the majority-rules solution re-use policy for SARTRE-2 and conducted a further set of experiments where SARTRE-2 challenged SARTRE-1. Table 5 presents the outcomes of 10 duplicate matches, where $N = 3000$. The figures presented are the small bets per hand SARTRE-2 won against SARTRE-1. In total 60,000 hands of poker were played between the two systems.

Table 5. Results of self play matches between SARTRE-2 and SARTRE-1. Each match consists of 6000 hands. Results for SARTRE-2 are recorded in **sb/h**.

Match	SARTRE-2 **(sb/h)**
1	−0.034
2	0.30
3	0.016
4	0.005
5	0.001
6	−0.013
7	−0.004
8	0.011
9	0.004
10	0.000
Mean:	**0.0286**
Std dev:	**0.096368**
95% CI:	**[-0.04034, 0.09754]**

Table 5 shows that for the 10 duplicate matches played, SARTRE-2 won on average 0.0286 sb/h against SARTRE-1. Out of the 10 matches SARTRE-2 won 6 and lost 3. The final match was a draw. A 95% confidence interval was calculated on the sample using a student's t-distribution. The result shows that while SARTRE-2 achieves a greater average profit than SARTRE-1, further evaluation is required to achieve statistical significance.

6 Discussion

The results indicate that simply re-using the decision made the majority of the time results in better performance than mixing from a probability triple and

that choosing the decision that resulted in the best outcome was the worst solution re-use policy. Moreover, while we have not presented results against other computerised agents, our initial testing appears to suggest the same outcomes are observed.

One of the reasons for the poor performance of **best-outcome** is likely due to the fact that good outcomes don't necessarily represent good betting decisions and vice-versa. The reason for the success of the **majority-rules** policy is less clear. We believe this has to do with the type of opponent being challenged, i.e. Nash equilibrium-based agents. As an equilibrium-based strategy doesn't attempt to exploit any bias in its opponent strategy, it will only gain when the opponent ends up making a mistake. Therefore, biasing actions towards the decision that was made the majority of the time is likely to go unpunished when challenging an equilibrium-based agent. Furthermore, sticking to this decision avoids any **exploration** errors made by choosing other actions. On the other hand, against an exploitive opponent the bias imposed by choosing only one action is likely to be detrimental to performance and therefore it would become more important to mix up decisions.

By modifying the way knowledge is encoded within the case-base knowledge container, SARTRE-2 allows a significant reduction in case-base size. In particular, SARTRE-2 stores 904,886 fewer cases than SARTRE-1. Futhermore, the results of experiment 2 (SARTRE-1 Vs. SARTRE-2) show that SARTRE-2 appears to perform a little better than SARTRE-1. Once again, initial results against other computerised agents (not included in the paper) appear to support this observation. There are many factors that could have contributed to this difference in performance, including training on different "expert" players, the updated betting sequence similarity metric, as well as the improved problem coverage of SARTRE-2 as a result of the updated solution representation.

7 Conclusions

We have presented the SARTRE system. Given hand log training data the SARTRE system stores the betting decisions in a case-base and generalises the observed decisions to inform a betting strategy. During game play, actions are chosen by searching the appropriate case-base and retrieving cases similar to the present situation. SARTRE selects a betting action based on 1 of 3 solution re-use policies. A version of the SARTRE system, which we have labelled SARTRE-1, was submitted to the 2009 IJCAI Computer Poker Competition. The competitors in this competition typically represent the current state-of-the-art in computer poker agents. In the 2009 competition, SARTRE placed 6th out of 12 in the limit bankroll event and 7th out of 13 in the equilibrium event. Given the simplistic nature of the approach the results are quite positive.

Case-base maintenance was applied to the system. In this paper we referred to the post-maintenance system as SARTRE-2. Observations made during the maintenance process provided support for claim 1 that *changes in solution representation results in changes to problem coverage.* In particular, updates to

SARTRE-2's solution representation resulted in the removal of duplicate cases from the case-base. Reducing the number of cases required for storage had the effect of increasing the amount of scenarios SARTRE-2 was able to encounter and use for training purposes. This led to a more comprehensive case-base being generated.

SARTRE-2's case representation is concise which allows a compact representation of a betting strategy for limit Texas Hold'em. The attribute-value pairs that make up SARTRE's case representation were described along with the betting sequence similarity metrics used by the system, which allow graceful degradation when an exactly matching case is unable to be retrieved.

The second claim investigated was that *different policies for re-using solutions leads to changes in performance*. Empirical results, presented in Table 4, were used to support this claim. Three *solution re-use policies* were introduced and comparatively evaluated. The results showed that the *majority-rules* policy achieved the greatest profit during self-play experiments. Given this result, SARTRE-1 was challenged by SARTRE-2 using a majority-rules decision re-use policy. The results showed that on average SARTRE-2 achieved a greater profit than SARTRE-1. However, as the systems are still closely related in strength, many more hands are required in order to display a significant difference.

8 Future Work

In the future we wish to introduce opponent modeling. Currently SARTRE attempts to approximate a robust betting strategy that is able to perform well against a range of opponents without paying any attention to how the opponent plays. We wish to investigate possible ways of augmenting the current system with opponent modeling information, which could be used to exploit weaker opponents.

References

1. Billings, D., Papp, D., Schaeffer, J., Szafron, D.: Poker as Testbed for AI Research. In: Advances in Artificial Intelligence, 12th Biennial Conference of the Canadian Society for Computational Studies of Intelligence, pp. 228–238. Springer (1998)
2. Rubin, J., Watson, I.: A Memory-Based Approach to Two-Player Texas Hold'em. In: AI 2009: Advances in Artificial Intelligence, 22nd Australasian Joint Conference, pp. 465–474. Springer (2009)
3. Rubin, J., Watson, I.: SARTRE: System Overview, A Case-Based Agent for Two-Player Texas Hold'em. In: Workshop on Case-Based Reasoning for Computer Games, Eighth International Conference on Case-Based Reasoning (ICCBR 2009). Springer (2009)
4. Leake, D, B., Wilson, D, C.: Categorizing Case-Base Maintenance: Dimensions and Directions. In: Advances in Case-Based Reasoning, 4th European Workshop (EWCBR-98). pp. 196–207. Springer (1998)
5. Texas Hold'em Article, `http://en.wikipedia.org/wiki/Texas_Hold'em`
6. Morris, P.: Introduction to game theory. Springer-Verlag, New York (1994)

7. Johanson, M, B.: Robust Strategies and Counter-Strategies: Building a Champion Level Computer Poker Player. Masters thesis, University of Alberta (2007)
8. Billings, D., Burch, N., Davidson, A., Holte, R.C., Schaeffer, J., Schauenberg, T., Szafron, D.: Approximating Game-Theoretic Optimal Strategies for Full-scale Poker. In: IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, pp. 661–668. Morgan Kaufmann (2003)
9. Gilpin, A., Sandholm, T.: A Competitive Texas Hold'em Poker Player via Automated Abstraction and Real-Time Equilibrium Computation. In: Proceedings, The Twenty-First National Conference on Artificial Intelligence (AAAI 2006), pp. 1007–1013. AAAI Press (2006)
10. Gilpin, A., Sandholm, T.: Better Automated Abstraction Techniques for Imperfect Information Games, with Application to Texas Hold'em Poker. In: 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), pp. 192–200. IFAAMAS (2007)
11. Zinkevich, M., Johanson, M., Bowling, M. H., Piccione, C.:Regret Minimization in Games with Incomplete Information. In: Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, pp. 1729-1736. MIT Press (2007)
12. 2009 Computer Poker Winners, `http://www.cs.ualberta.ca/~pokert/2009/results/winners.txt`
13. Billings, D., Davidson, A., Schauenberg, T., Burch, N., Bowling, M., Holte, R.C., Schaeffer, J., Szafron, D.: Game-Tree Search with Adaptation in Stochastic Imperfect-Information Games. In: Computers and Games, 4th International Conference, CG 2004, pp. 21–34. Springer (2004)
14. Billings, D., Castillo, L, P., Schaeffer, J., Szafron, D.: Using Probabilistic Knowledge and Simulation to Play Poker. In: AAAI/IAAI, pp. 697–703. AAAI Press (1999)
15. Schweizer, I., Panitzek, K., Park, S, H., Furnkranz, J.: An Exploitative Monte-Carlo Poker Agent. Technical report, Technische Universitat Darmstadt (2009)
16. Van den Broeck, G., Driessens, K., Ramon, J.: Monte-Carlo Tree Search in Poker Using Expected Reward Distributions. In: Advances in Machine Learning, First Asian Conference on Machine Learning (ACML 2009), pp. 367–381. Springer (2009)
17. Johanson, M., Zinkevich, M., Bowling, M, H.: Computing Robust Counter-Strategies. In: Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, pp. 721–728. MIT Press (2007)
18. Johanson, M., Bowling, M.: Data Biased Robust Counter Strategies. In: Twelfth International Conference on Artificial Intelligence and Statistics, pp. 264–271. (2009)
19. Rubin, J., Watson, I.: Investigating the Effectiveness of Applying Case-Based Reasoning to the Game of Texas Hold'em. In: Proceedings of the Twentieth International Florida Artificial Intelligence Research Society Conference, pp. 417–422. AAAI Press (2007)
20. Watson, I., Rubin, J.: CASPER: A Case-Based Poker-Bot. In: AI 2008: Advances in Artificial Intelligence, 21st Australasian Joint Conference on Artificial Intelligence, pp. 594–600. Springer (2008)
21. 2008 Poker Bot Competition Summary, `http://www.cs.ualberta.ca/~pokert/2008/results/`
22. 2009 Poker Bot Competition Summary, `http://www.cs.ualberta.ca/~pokert/2009/results/`