Case-Based Strategies in Computer Poker

Jonathan Rubin [a] and Ian Watson [a]
[a] *Department of Computer Science. University of Auckland Game AI Group*
E-mail: jrubin01@gmail.com,
E-mail: ian@cs.auckland.ac.nz

The state-of-the-art within Artificial Intelligence has directly benefited from research conducted within the computer poker domain. One such success has been the advancement of *bottom up* equilibrium finding algorithms via computational game theory. On the other hand, alternative *top down* approaches, that attempt to generalise decisions observed within a collection of data, have not received as much attention. In this work we employ a *top down* approach in order to construct *case-based strategies* within three computer poker domains. Our analysis begins within the simplest variation of Texas Hold'em poker, i.e. two-player, limit Hold'em. We trace the evolution of our *case-based* architecture and evaluate the effect that modifications have on strategy performance. The end result of our experimentation is a coherent framework for producing strong *case-based strategies* based on the observation and generalisation of expert decisions. The lessons learned within this domain offer valuable insights, that we use to apply the framework to the more complicated domains of two-player, no-limit Hold'em and multi-player, limit Hold'em. For each domain we present results obtained from the Annual Computer Poker Competition, where the best poker agents in the world are challenged against each other. We also present results against human opposition.

Keywords: Imperfect Information Games, Game AI, Case-Based Reasoning

1. Introduction

The state-of-the-art within Artificial Intelligence (AI) research has directly benefited from research conducted within the computer poker domain. Perhaps its most notable achievement has been the advancement of equilibrium finding algorithms via computational game theory. State-of-the-art equilibrium finding algorithms are now able to solve mathematical models that were once prohibitively large. Furthermore, empirical results tend to support the intuition that solving larger models results in better quality strategies[1]. However, equilibrium finding algorithms are only one of many approaches available within the computer poker test-bed. Alternative approaches such as imperfect information game tree search [8] and, more recently, Monte-Carlo tree search [36] have also received attention from researchers in order to handle challenges within the computer poker domain that cannot be suitably addressed by equilibrium finding algorithms, such as dynamic adaptation to changing game conditions.

The algorithms mentioned above take a *bottom up* approach to constructing sophisticated strategies within the computer poker domain. While the details of each algorithm differ, they roughly achieve their goal by enumerating (or sampling) a state space together with its pay-off values in order to identify a distribution over actions that achieves the greatest *expected value*. An alternative *top down* procedure attempts to construct sophisticated strategies by generalising decisions observed within a collection of data. This *lazier* top down approach offers its own set of problems in the domain of computer poker. In particular, any top down approach is a slave to its data, so quality data is a necessity. While massive amounts of data from online poker sites is available [25], the quality of the decisions contained within this data is usually questionable. The imperfect information world of the poker domain can often mean that valuable information may be missing from this data. Moreover, the stochastic nature of the poker domain ensures that it is not enough to simply rely on outcome information in order to determine decision quality.

Despite the problems described above, *top down* approaches within the computer poker domain have still managed to produce strong strategies [4,28]. In fact, empirical evidence from interna-

[1] See [38] for a discussion of why this is not always the case.

AI Communications 25 (2012) 1948
DOI 10.3233/AIC-2012-0513
ISSN 0921-7126,

tional computer poker competitions [1] suggest that, in a few cases, top down approaches have managed to out-perform their bottom up counterparts. In this work we describe one such *top down* approach that we have used to construct sophisticated strategies within the computer poker domain. Our *case-based approach* can be used to produce strategies for a range of sub-domains within the computer poker environment, including both limit and no-limit betting structures as well as two-player and multi-player matches. The *case-based strategies* produced by our approach have achieved 1^{st} place finishes for our agent (Sartre) at the Annual Computer Poker Competition (ACPC) [1]. The ACPC is the premier computer poker event and the agents submitted typically represent the current state-of-the-art in computer poker research.

We have applied and evaluated case-based strategies within the game of Texas Hold'em. Texas Hold'em is currently the most popular poker variation. To achieve strong performance, players must be able to successfully deal with imperfect information, i.e. they cannot see their opponents' hidden cards. Also, chance events occur in the domain via the random distribution of playing cards. Texas Hold'em can be played as a two-person game or a multi-player game. There are multiple variations on the type of betting structures used that can dramatically alter the dynamics of the game and hence the strategies that must be employed for successful play. For instance, a **limit** game restricts the size of the bets allowed to predefined values. On the other hand, a **no-limit** game imposes no such restriction.

In this work we present case-based strategies in three poker domains. Our analysis begins within the simplest variation of Texas Hold'em, i.e. two-player, limit Hold'em. Here we trace the evolution of our case-based architecture and evaluate the effect that modifications have on strategy performance. The end result of our experimentation in the two-player, limit Hold'em domain is a coherent framework for producing strong case-based strategies, based on the observation and generalisation of expert decisions. The lessons learned within this domain offer valuable insights, which we use to apply the framework to the more complicated domains of two-player, no-limit Hold'em and multi-player, limit Hold'em. We describe the difficulties that these more complicated domains impose and how our framework deals with these issues. For each of the three poker sub-domains mentioned above we produce strategies that have been extensively evaluated. In particular, we present results from Annual Computer Poker Competitions for the years 2009 – 2011 and illustrate the performance trajectory of our case-based strategies against the best available opposition.

The remainder of this document proceeds as follows. Section 2 describes the rules of Texas Hold'em poker, highlighting the differences between the different variations available. Section 3 provides the necessary background and details some related work. Section 4 further recaps the benefits of the poker domain as a test-bed for artificial intelligence research and provides the motivation for the use of case-based strategies as opposed to alternative algorithms. Section 5 details the initial evolution of our case-based architecture for computer poker in the two-player, limit Hold'em domain. Experimental results are presented and discussed. Sections 6 and 7 extrapolate the resulting framework to the more complicated domains of two-player, no-limit Hold'em and multi-player limit Hold'em. Once again, results are presented and discussed for each separate domain. Finally, Section 8 concludes the document.

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 more detailed information on Texas Hold'em consult [33], or for further information on poker in general see [32].

Texas Hold'em can be played either as a two-player game or a **multi-player** game. When a game consists only of two players it is often referred to as a **heads up** match. Game play consists of four stages – **preflop**, **flop**, **turn** and **river**. During each stage a round of betting occurs. The first round of play is the preflop where 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. In a heads up match, the dealer acts first preflop. In a multi-player match the player to the left of the big blind acts first pre-

flop. In both heads up and multi-player matches, the dealer is the last to act on the post-flop betting rounds (i.e. the flop, turn and river). The legal betting actions are fold, check/call or bet/raise. These possible betting actions are common to all variations of poker and are described in more detail below:

Fold: When a player contributes no further chips to the pot and abandons their hand and any right to contest the chips that 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 continues 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 a player may bet any amount up to the total value of chips that they possess. For example, assuming a player begins a match with 1000 chips, after paying a forced small blind of one chip they then have the option to either fold, call one more chip or raise by contributing anywhere between 3 and 999 extra chips[2]. In a standard game of heads-up, no-limit poker, both players' chip stacks would fluctuate between hands, e.g. a win from a previous hand would ensure that one player had a larger chip stack to play with on the next hand. In order to reduce the variance that this structure imposes, a variation known as **Doyle's Game** is played where the starting stacks of both players are reset to a specified amount at the beginning of every hand.

Once the round of betting is complete, as long as at least two players still remain in the hand, play continues on to the next stage. Each post-flop stage involves the drawing of **community cards** from the shuffled deck of cards as follows: **flop** – 3 community cards, **turn** – 1 community card, **river** – 1 community card. All players combine their hole cards with the public community cards to form their best five card poker hand. 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 both players' hands are of equal value, the pot is split between them.

[2]The minimum raise would involve paying 1 more chip to match the big blind and then committing at least another 2 chips as the minimum legal raise.

3. Background

3.1. Strategy Types

As mentioned in the introduction, many AI researchers working in the computer poker domain have focused their efforts on creating strong strategies via *bottom up*, equilibrium finding algorithms. When equilibrium finding algorithms are applied to the computer poker domain, they produce ϵ-Nash equilibria. **ϵ-Nash equilibria** are robust, static strategies that limit their exploitability (ϵ) against worst-case opponents. A pair of strategies are said to be an ϵ-Nash equilibrium if neither strategy can gain more than ϵ by deviating. In this context, a strategy refers to a probabilistic distribution over available actions at every decision point. Two state-of-the-art equilibrium finding algorithms are Counterfactual Regret Minimisation (CFRM) [39,18] and Excessive Gap Technique (EGT) [13]. CFRM is an iterative, regret minimising algorithm that was developed by the University of Alberta Computer Poker Research Group (CPRG)[3]. The EGT algorithm, developed by Andrew Gilpin and Thomas Sandholm from Carnegie Mellon University, is an adapted version of Nesterov's *excessive gap technique* [21], which has been specialised for two-player, zero-sum, imperfect information games.

The ϵ-Nash equilibrium strategies produced via CFRM and EGT are solid, unwavering strategies that do not adapt given further observations made by challenging particular opponents. An alternative strategy type is one that attempts to exploit perceived weaknesses in their opponents' strategies, by dynamically adapting their strategy given further observations. This type of strat-

[3]http://poker.cs.ualberta.ca/

egy is known as an **exploitive** (or **maximal**) strategy. Exploitive strategies typically select their actions based on information they have observed about their opponent. Therefore, constructing an exploitive strategy typically involves the added difficulty of generating accurate opponent models.

3.2. Strategy Evaluation and the Annual Computer Poker Competition

Both ϵ-Nash equilibrium based strategies and exploitive strategies have received attention in the computer poker literature [14,15,7,8,17]. Overall a larger focus has been applied to equilibrium finding approaches. This is especially true regarding agents entered into the Annual Computer Poker Competition. Since 2006, the ACPC has been held every year at conferences such as AAAI and IJCAI. The agents submitted to the competition typically represent the strongest computer poker agents in the world, for that particular year. Since 2009, the ACPC has evaluated agents in the following variations of Texas Hold'em:

1. Two-player, Limit Hold'em.
2. Two-player, No-Limit Hold'em.
3. Three-player, Limit Hold'em.

In this work, we restrict our attention to these three sub-domains. Agents are evaluated by playing many hands against each other in a round-robin tournament structure. The ACPC employs two winner determination procedures:

1. **Total Bankroll.** As its name implies the **total bankroll** winner determination simply records the overall profit or loss of each agent and uses this to rank competitors. In this division, agents that are able to achieve larger bankrolls are ranked higher than those with lower profits. This winner determination procedure does not take into account how an agent achieves its overall profit or loss, for instance it is possible that the winning agent could win a large amount against one competitor, but lose to all other competitors.
2. **Bankroll Instant Run-Off.** On the other hand, the **instant run-off** division uses a recursive winner determination algorithm that repeatedly removes the agents that performed the worst against a current pool of players. This way agents that achieve large profits by exploiting weak opponents are not favoured, as in the **total bankroll** division.

As poker is a stochastic game that consists of chance events, the variance can often be large especially between agents that are close in strength. This requires many hands to be played in order to arrive at statistically significant conclusions. Due to the large variance involved, the ACPC employs a **duplicate match** structure, whereby all players end up playing the same set of hands. For example, in a two-player match a set of N hands are played. This is then followed by dealing the same set of N hands a second time, but having both players switch seats so that they receive the cards their opponent received previously. As both players are exposed to the same set of hands, this reduces the amount of variance involved in the game by ensuring one player does not receive a larger proportion of higher quality hands than the other. A two-player match involves two seat enumerations, whereas a three-player duplicate match involves six seat enumerations to ensure each player is exposed to the same scenario as their opponents. For three players (ABC) the following seat enumerations need to take place:

ABC ACB
CAB CBA
BCA BAC

4. Research Motivation

This work describes the use of *case-based* strategies in games. Our approach makes use of the Case-based Reasoning (CBR) methodology [26,19]. The CBR methodology encodes problems, and their solutions, as cases. CBR attempts to solve new problems or scenarios by locating similar past problems and re-using or adapting their solutions for the current situation. Case-based strategies are top down strategies, in that they are constructed by processing and analysing a set of training data. Common game scenarios, together with their playing decisions are captured as a collection of cases, referred to as the *case-base*. Each case attempts to capture important game state information that is likely to have an impact on the final playing decision. The training data can be both real-world data, e.g. from online poker casinos, or artificially generated data, for instance from hand history logs generated by the ACPC. Case-based strategies attempt to generalise the game playing deci-

sions recorded within the data via the use of similarity metrics that determine whether two game playing scenarios are sufficiently similar to each other, such that their decisions can be re-used.

Case-based strategies can be created by training on data generated from a range of expert players or by isolating the decisions of a single expert player. Where a case-based strategy is produced by training on and generalising the decisions of a single expert player, we refer to the agent produced as an *expert imitator*. In this way, case-based strategies can be produced that attempt to imitate different styles of play simply by training on separate datasets generated by observing the decisions of *expert* players, each with their own style. The lazy learning [2] of case-based reasoning is particularly suited to expert imitation where observations of expert play can be recorded and stored for use at decision time.

Case-based approaches have been applied and evaluated in a variety of gaming environments. CHEBR [24] was a case-based checkers player that acquired experience by simply playing games of checkers in real-time. In the RoboCup soccer domain, [11] used case-based reasoning to construct a team of agents that observes and imitates the behaviour of other agents. Case-based planning [16] has been investigated and evaluated in the domain of real-time strategy games [3,22,23,34]. Case-based tactician (CaT) described in [3] selects tactics based on a state lattice and the outcome of performing the chosen tactic. The CaT system was shown to successfully learn over time. The Darmok architecture described by [22,23] pieces together fragments of plans in order to produce an overall playing strategy. Performance of the strategies produced by the Darmok architecture were improved by first classifying the situation it found itself in and having this affect plan retrieval [20]. Combining CBR with other AI approaches has also produced successful results. In [31] transfer learning was investigated in a real time strategy game environment by merging CBR with reinforcement learning. Also, [6] combined CBR with reinforcement learning to produce an agent that could respond rapidly to changes in conditions of a domination game.

The stochastic, imperfect information world of Texas Hold'em poker is used as a test-bed to evaluate and analyse our case-based strategies. Texas Hold'em offers a rich environment that allows the opportunity to apply an abundance of strategies ranging from basic concepts to sophisticated strategies and counter-strategies. Moreover, the rules of Texas Hold'em poker are incredibly simple. Contrast this with CBR related research into complex environments such as real-time strategy games [3,20,22,23], which offer similar issues to deal with – uncertainty, chance, deception – but don't encapsulate this within a simple set of rules, boundaries and performance metrics. Successes and failures achieved by applying case-based strategies to the game of poker may provide valuable insights for CBR researchers using complex strategy games as their domain, where immediate success is harder to evaluate. Furthermore, it is hoped that results may also generalise to domains outside the range of games altogether to complex real world domains where hidden information, chance and deception are commonplace.

One of the major benefits of using case-based strategies within the domain of computer poker is the simplicity of the approach. Top down case-based strategies don't require the construction of massive, complex mathematical models that some other approaches rely on [13,30,27]. Instead, an autonomous agent can be created simply via the observation of expert play and the encoding of observed actions into cases. Below we outline some further reasons why case-based strategies are suited to the domain of computer poker and hence worthy of investigation. The reasons listed are loosely based on Sycara's [35] identification of characteristics of a domain where case-based reasoning is most applicable (these were later adjusted by [37]).

1. **A case is easily defined in the domain.**
   A case is easily identified as a previous scenario an (expert) player has encountered in the past and the action (solution) associated with that scenario such as whether to fold, call or raise. Each case can also record a final outcome from the hand, i.e. how many chips a player won or lost.
2. **Expert human poker players compare current problems to past cases.**
   It makes sense that poker experts make their decisions based on experience. An expert poker player will normally have played many games and encountered many different scenarios; they can then draw on this experience to determine what action to take for a current problem.

3. **Cases are available as training data.** While many cases are available to train a case-based strategy, the quality of their solutions can vary considerably. The context of the past problem needs to be taken into account and applied to similar contexts in the future. As the system gathers more experience it can also record its own cases, together with their observed outcomes.
4. **Case comparisons can be done effectively.** Cases are compared by determining the similarity of their local features. There are many features that can be chosen to represent a case. Many of the salient features in the poker domain (e.g. *hand strength*) are easily comparable via standard metrics. Other features, such as *betting history*, require more involved similarity metrics, but are still directly comparable.
5. **Solutions can be generalised.** For case-based strategies to be successful, the re-use or adaptation of similar cases' solutions should produce a solution that is (reasonably) similar to the actual, known solution (if one exists) of the target case in question. This underpins one of CBR's main assumptions: that similar cases have similar solutions. We present empirical evidence that suggests the above assumption is reasonable in the computer poker domain.

5. Two-Player, Limit Texas Hold'em

We begin with the application of case-based strategies within the domain of two-player, limit Texas Hold'em. Two-player, limit Hold'em offers a beneficial starting point for the experimentation and evaluation of case-based strategies, within computer poker. Play is limited to two players and a restricted betting structure is imposed, whereby all bets and raises are limited to pre-specified amounts. The above restrictions limit the size of the state space, compared to Hold'em variations that allow no-limit betting and multiple opponents. However, while the size of the domain is reduced, compared to more complex poker domains, the two-player limit Hold'em domain is still very large. The game tree consists of approximately 10^{18} game states and, given the standards of current hardware, it is intractable to derive a true Nash equilibrium for the game. In fact, it proves impossible to reasonably store this strategy by today's hardware standards [18]. For these reasons alternative approaches, such as case-based strategies, can prove useful given their ability for generalisation.

Over the years we have conducted an extensive amount of experimentation on the use of case-based strategies, using two-player, limit Hold'em as our test-bed. In particular we have investigated and measured the effect that changes have on areas such as *feature and solution representation*, *similarity metrics*, *system training* and the use of different *decision making policies*. Modifications have ranged from the very minor, e.g. training on different sets of data to the more dramatic, e.g. the development of custom betting sequence similarity metrics. For each modification and addition to the architecture we have extensively evaluated the strategies produced via self-play experiments, as well as by challenging a range of third-party, artificial agents and human opposition. Due to space limitations we restrict our attention to the changes that had the greatest affect on the system architecture and its performance. We have named our system Sartre (Similarity Assessment Reasoning for Texas hold'em via Recall of Experience) and we trace the evolution of its architecture below.

5.1. Overview

In order to generalise betting decisions from a set of (artificial or real-world) training data, first it is required to construct and store a collection of cases. A case's feature and solution representation must be decided upon, such as the identification of salient attribute-value pairs that describe the environment at the time a case was recorded. Each case should attempt to capture important information about the current environment that is likely to have an impact on the final solution. After a collection of cases has been established, decisions can be made by searching the case-base and locating similar scenarios for which solutions have been recorded in the past. This requires the use of local similarity metrics for each feature.

Given a target case, t, that describes the immediate game environment, a source case, $s \in S$, where S is the entire collection of previously recorded cases and a set of features, F, global similarity is computed by summing each feature's lo-

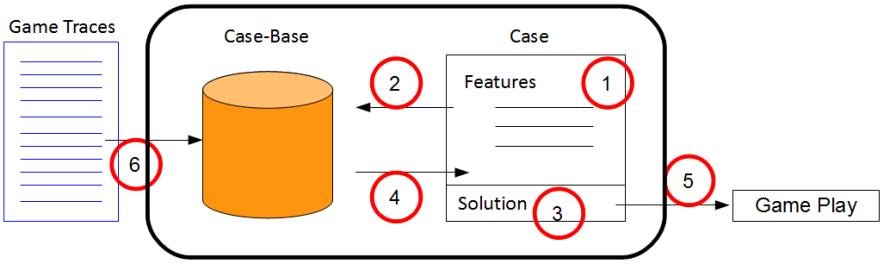

Fig. 1. Overview of the architecture used to produce case-based strategies. The numbers identify the six key areas within the architecture where the affects of maintenance has been evaluated.

cal similarity contribution, sim_f, and dividing by the total number of features:

$$G(t,s) = \sum_{f \in F} \frac{sim_f(t_f, s_f)}{|F|} \quad (1)$$

Fig. 1. provides a pictorial representation of the architecture we have used to produce case-based strategies. The six areas that have been labelled in Fig. 1. identify six key areas within the architecture where *maintenance* has had the most impact and led to positive affects on system performance. They are:

1. Feature Representation
2. Similarity Metrics
3. Solution Representation
4. Case Retrieval
5. Solution Re-Use Policies, and
6. System Training

5.2. Architecture Evolution

Here we describe some of the changes that have taken place within the six key areas of our case-based architecture, identified above. Where possible, we provide a comparative evaluation for the maintenance performed, in order to measure the impact that changes had on the performance of the case-based strategies produced.

5.2.1. Feature Representation

The first area of the system architecture that we discuss is the feature representation used within a case (see Fig. 1, Point 1). We highlight results that have influenced changes to the representation over time. In order to construct a case-based strategy a case representation is required that establishes the type of information that will be recorded for each game scenario. Our case-based strategies use a simple attribute-value representation to describe a set of case features. Table 1 lists the features used within our case representation. A separate representation is used for preflop and postflop cases, given the differences between these two stages of the game. The features listed in Table 1 were chosen by the authors as they concisely capture all the necessary public game information, as well as the player's personal, hidden information.

Table 1
Preflop and postflop case feature representation.

	Preflop	**Postflop**
1.	*Hole Cards*	*Hand Strength*
2.	*Betting Sequence*	*Betting Sequence*
3.		*Board Texture*

Each feature is explained in more detail below:

Preflop

1. **Hole Cards:** the personal hidden cards of the player, represented by 1 out of 169 equivalence classes.
2. **Betting Sequence:** a sequence of characters that represent the betting actions witnessed until the current decision point, where actions can be selected from the set, $A_{limit} = \{f, c, r\}$.

Postflop

1. **Hand Strength:** a description of the player's hand strength given a combination of their personal cards and the public community cards.
2. **Betting Sequence:** identical to the preflop sequence, however with the addition of round delimiters to distinguish betting from previous rounds, $A_{limit} \cup \{-\}$.

3. Board Texture: a description of the public community cards that are revealed during the postflop rounds

While the case features themselves have remained relatively unchanged throughout the architecture's evolution, the actual values that each feature records has been experimented with to determine the affect on final performance. For example, we have compared and evaluated the use of different metrics for the **hand strength** feature from Table 1. Fig. 2. depicts the result of a comparison between three **hand strength** feature values. In this experiment, the feature values for *betting sequence* and *board texture* were held constant, while the *hand strength* value was varied. The values used to represent **hand strength** were as follows:

CATEGORIES: Uses expert defined categories to classify hand strength. Hands are assigned into categories by mapping a player's personal cards and the available board cards into one of a number of predefined categories. Each category represents the type of hand the player currently has, together with information about the *drawing* potential of the hand, i.e. whether the hand has the ability to improve with future community cards. In total 284 categories were defined[4].

E[HS]: *Expected hand strength* is a one-dimensional, numerical metric. The E[HS] metric computes the probability of winning at showdown against a random hand. This is given by enumerating all possible combinations of community cards and determining the proportion of the time the player's hand wins against the set of all possible opponent holdings. Given the large variety of values that can be produced by the E[HS] metric, *bucketing* takes place where similar values are mapped into a discrete set of buckets that contain hands of similar strength. Here we use a total of 20 buckets for each postflop round.

E[HS^2]: The final metric evaluated involves squaring the *expected hand strength*. Johanson [18] points out that squaring the expected hand strength (E[HS^2]) typically gives better results, as this assigns higher hand strength values to hands with greater potential. Typically in poker, hands with similar strength values, but differences in potential, are required to be played in strategically different ways [33]. Once again bucketing is used where the derived E[HS^2] values are mapped into 1 of 20 unique buckets for each postflop round.

[4]A listing of all 284 categories can be found at the following website: `http://www.cs.auckland.ac.nz/research/gameai/sartreinfo.html`

The resulting case-based strategies were evaluated by challenging the computerised opponent Fell Omen 2 [10]. Fell Omen 2 is a solid two-player limit Hold'em agent that plays an ϵ-Nash equilibrium type strategy. Fell Omen 2 was made publicly available by its creator Ian Fellows and has become widely used as an agent for strategy evaluation [12]. The results depicted in Fig. 2. are measured in small bets per hand (sb/h), i.e. where the total number of small bets won or lost are divided by the total number of hands played. Each data point records the outcome of three matches, where 3000 duplicate hands were played. The 95% confidence intervals for each data point are also shown.

Results were recorded for various levels of case-base usage to get an idea of how well the system is able to generalise decisions. The results in Fig. 2. show that (when using a full case-base) the use of E[HS^2] for the hand strength feature produces the strongest strategies, followed by the use of CATEGORIES and finally E[HS]. The poor performance of E[HS] is likely due to the fact that this metric does not fully capture the importance of a hand's future *potential*. When only a partial proportion of the case-base is used it becomes more important for the system to be able to recognise similar attribute values in order to make appropriate decisions. Both E[HS] and E[HS^2] are able to generalise well. However, the results show that decision generalisation begins to break down when using CATEGORIES. This has to do with the similarity metrics used. In particular, the CATEGORIES strategy in Fig. 2 is actually a baseline strategy that used overly simplified similarity metrics for each of its feature values. Next we discuss the area of *similarity assessment* within the system architecture, which is intimately tied to the particular values chosen within the feature representation.

5.2.2. Similarity Assessment

For each feature that is used to represent a case, a corresponding local similarity metric, $sim_f(f_1, f_2)$, is required that determines how similar two feature values, f_1 and f_2, are to each other.

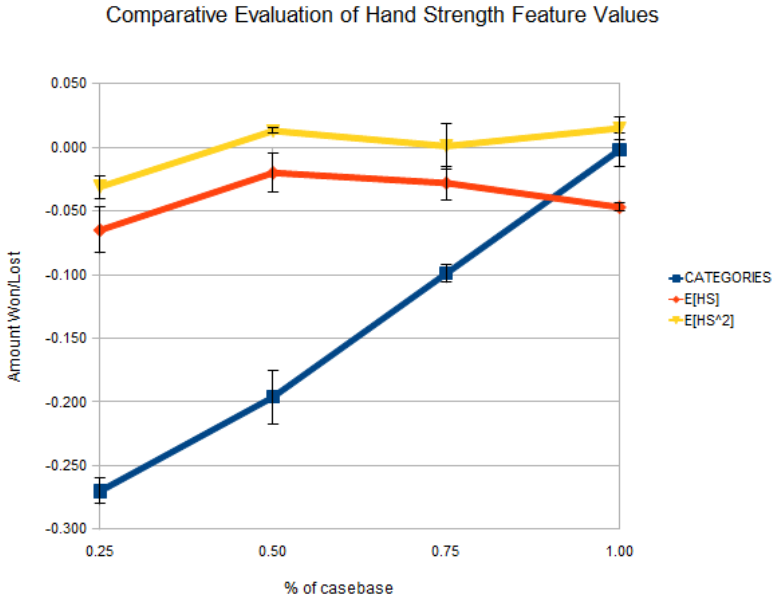

Fig. 2. The performance of three separate case-based strategies produced by altering the value used to represent hand strength. Results are measured in sb/h and were obtained by challenging Fell Omen 2.

The use of different representations for the hand strength feature in Fig. 2. also requires the use of separate similarity metrics. The CATEGORIES strategy in Fig. 2. employs a trivial *all-or-nothing* similarity metric for each of its features. If the value of one feature has the same value of another feature, a similarity score of 1 is assigned. On the other hand, if the two feature values differ at all, a similarity value of 0 is assigned. This was done to get an initial idea of how the system performed using the most basic of similarity retrieval measures. The performance of this baseline system could then be used to determine how improvements to local similarity metrics affected overall performance.

The degradation of performance observed in Fig. 2. for the CATEGORIES strategy (as the proportion of case-base usage decreases) is due to the use of *all-or-nothing* similarity assessment. The use of the overly simplified *all-or-nothing* similarity metric meant that the system's ability to retrieve similar cases could often fail, leaving the system without a solution for the current game state. When this occurred a *default-policy* was relied upon to provide the system with an action. The default-policy used by the system was an *always-call* policy, whereby the system would first attempt to check if possible, otherwise it would call an opponent's bet. This default-policy was selected by the authors as it was believed to be preferable to other trivial default policies, such as *always-fold*, which would always result in a loss for the system.

The other two strategies in Fig. 2. (E[HS] and E[HS^2]) do not use trivial *all-or-nothing* similarity. Instead the *hand strength* features use a similarity metric based on Euclidean distance. Both the E[HS] and E[HS^2] strategies also employ informed similarity metrics for their *betting sequence* and *board texture* features, as well. Recall that the *betting sequence* feature is represented as a sequence of characters that lists the playing decisions that have been witnessed so far for the current hand. This requires the use of a non-trivial metric to determine similarity between two non-identical sequences. Here we developed a custom similarity metric that involves the identification of *stepped* levels of similarity, based on the number of bets/raises made by each player. The exact details of this metric are presented in Section 5.3.2. Finally, for completeness, we determine similarity between different *board texture* classes via the use of hand picked similarity values.

Fig. 2. shows that, compared to the CATEGORIES strategy, the E[HS] and E[HS^2] strategies

do a much better job of decision generalisation as the usable portion of the case-base is reduced. The eventual strategies produced do not suffer the dramatic performance degradation that occurs with the use of *all-or-nothing* similarity.

5.2.3. Solution Representation

As well as recording feature values, each case also needs to specify a solution. The most obvious solution representation is a single betting action, $a \in A_{limit}$. As well as a betting action, the solution can also record the actual outcome, i.e. the numerical result, $o \in \Re$, of having taken action $a \in A_{limit}$, for a particular hand. Using this representation allows a set of training data to be parsed where each action/outcome observed maps directly to a case, which is then stored in the case-base. Case retention during game play can also be handled in the same way, where the case-base is updated with new cases at runtime. To make a playing decision at runtime the case-base is searched for similar cases and their solutions are combined to give a probabilistic vector (f, c, r) that specifies the proportion of the time our strategy should take each particular action.

A major drawback of the single action/outcome solution representation is that the case-base becomes filled with redundant cases, i.e. cases that record the same feature values, which also may or may not record the same betting action. An alternative solution representation, would involve directly storing the action and outcome vectors, i.e. effectively merging the redundant cases into one case by shifting combined information into the case solution. In order to achieve this solution representation, it now becomes necessary to pre-process the training data. As there is no longer a one-to-one mapping between hands and cases, the case-base must first be searched for an existing case during case-base construction. If one exists, its solution is updated, if one doesn't exist, it is added to the case-base. Case retention at runtime can also still take place, however, this requires that case solution vectors not be normalised after case-base construction as this would result in a loss of probabilistic information. Instead, it is sufficient to store the raw action/outcome frequency count information as this can later be normalised at runtime to make a probabilistic decision.

A solution representation based on action and outcome vectors results in a much more compact case-base. Table 2 depicts how much we were able to decrease the number of cases required to be stored simply by modifying the solution representation. Table 2 indicates that a single valued solution representation requires over 10 times the number of cases to be stored compared to a vector valued representation. Moreover, no information is lost when switching from a single valued representation to a vector valued representation. This modification has a follow on effect that improves the quality of the case-based strategies produced. As the number of cases required to be stored is so large – given the single action/outcome representation, training of the system had to be cut short due to the costs involved in actually storing the cases. The compact case-base produced by representing solutions directly as vectors, bypasses this problem and allows the system to be trained on a larger set of data. By not prematurely restricting the training phase, the case-based strategies produced are able to increase the amount of scenarios they are able to encounter and encode cases for during training. This leads to a more comprehensive and competent case-base.

Table 2
Total cases stored for each playing round using single value solution representation compared to vector valued solutions

Round	Total Cases - Single	Total Cases - Vector
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

5.2.4. Case Retrieval

Our architecture for producing case-based strategies has consistently used the k-nearest neighbour algorithm (k-NN) for case retrieval. Given a target case, t, and a collection of cases, S, the k-NN algorithm retrieves the k most similar cases by positioning t in the n-dimensional search space of S. Each dimension in the space records a value for one of the case features. Equation 1 (from Section 5.1) is used to determine the global similarity between two cases, t and $s \in S$.

While the use of the k-NN algorithm for case retrieval has not changed within our architecture, maintenance to other components of the architecture has resulted in modifications regarding the exact details used by the k-NN algorithm. One such modification was required given the transi-

tion from a single action solution representation to a vector valued solution representation (as described in Section 5.2.3). Initially, a variable value of k was allowed, whereby the total number of similar cases retrieved varied with each search of the case-base. Recall, that a case representation that encodes solutions as single actions results in a redundant case-base that contains multiple cases with the exact same feature values. The solution of those cases may or may not advocate different playing decisions. Given this representation, a final probability vector was required to be created on-the-fly at runtime by retrieving all identical cases and merging their solutions. Hence, the number of retrieved cases, k, could vary between 0 and N. When $k > 0$, the normalised entries of the probability vector were used to make a final playing decision. However, if $k = 0$, the *always-call* default-policy was used.

Once the solution representation was updated to record action vectors (instead of single decisions) a variable k value was no longer required. Instead, the algorithm was updated to simply always retrieve the nearest neighbour, i.e. $k = 1$. Given further improvements to the similarity metrics used, the use of a *default-policy* was no longer required as it was no longer possible to encounter scenarios where no similar cases could be retrieved. Instead, the most similar neighbour was always returned, no matter what the similarity value. This has resulted in a much more robust system that is actually capable of generalising decisions recorded in the training data, as opposed to the initial prototype system which offered no ability for graceful degradation, given dissimilar case retrieval.

5.2.5. Solution Re-use Policies

The fifth area of the architecture that we discuss (Fig. 1, Point 5) concerns the choice of a final playing decision via the use of separate policies, given a retrieved case and its solution. Consider the probabilistic action vector, $A = (a_1, a_2, \ldots, a_n)$, and a corresponding outcome vector, $O = (o_1, o_2, \ldots, o_n)$. There are various ways to use the information contained in the vectors to make a final playing decision. We have identified and empirically evaluated several different policies for re-using decision information, which we label **solution re-use policies**. Below we outline three solution re-use policies, which have been used for making final decisions by our case-based strategies.

1. **Probabilistic** The first solution re-use policy simply selects a betting action probabilistically, given the proportions specified within the action vector, $P(a_i) = a_i$, for $i = 1 \ldots n$. Betting decisions that have greater proportions within the vector will be made more often then those with lower proportions. In a game-theoretic sense, this policy corresponds to a mixed strategy.
2. **Max-frequency** Given an action vector $A = (a_1, a_2, \ldots, a_n)$, the *max-frequency* solution re-use policy selects the action that corresponds to $\arg\max_i a_i$, i.e. it selects the action that was made most often and ignores all other actions. In a game-theoretic sense, this policy corresponds to a pure strategy.
3. **Best-Outcome** Instead of using the values contained within the action vector, the *best-outcome* solution re-use policy selects an action, given the values contained within the outcome vector, $O = (o_1, o_2, \ldots, o_n)$. The final playing decision is given by the action, a_i, that corresponds to $\arg\max_i o_i$, i.e. the action that corresponds to the maximum entry in the outcome vector.

Given the three solution re-use policies described above, it is desirable to know which policies produce the strongest strategies. Table 3 presents the results of self-play experiments where the three solution re-use policies were challenged against each other. A round robin tournament structure was used, where each policy challenged every other policy. The figures presented are from the row player's perspective and are in small bets per hand. Each match consists of 3 separate duplicate matches of 3000 hands. Hence, in total 18,000 hands of poker were played between each competitor. All results are statistically significant with a 95% level of confidence.

Table 3 shows that the *max-frequency* policy outperforms its *probabilistic* and *best-outcome* counterparts. Of the three, best-outcome fares the worst, losing all of its matches. The results indicate that simply re-using the most commonly made decision results in better performance than mixing from a probability vector and that choosing the decision that resulted in the best outcome was the worst solution re-use policy. Moreover, these results are representative of further experiments involving other third-party computerised agents.

Table 3
Results of experiments between solution re-use policies. The values shown are in sb/h with 95% confidence intervals.

	Max-frequency	Probabilistic	Best-outcome	**Average**
Max-frequency		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		$\mathbf{-0.056 \pm 0.005}$

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 *max-frequency* policy is less obvious. In our opinion, this has to do with the type of opponent being challenged, i.e. agents that play a static, non-exploitive strategy, such as those listed in Table 3, as well as strategies that attempt to approximate a Nash equilibrium. As an equilibrium-based strategy does not attempt to exploit any bias in its opponent's strategy, it will only gain when the opponent ends up making a mistake by selecting an inappropriate action. The action that was made most often is unlikely to be an inappropriate action, therefore sticking to this decision avoids any exploration errors made by choosing other (possibly inappropriate) actions. Moreover, biasing playing decisions towards this action is likely to go unpunished when challenging a non-exploitive agent. On the other hand, against an exploitive opponent the bias imposed by choosing only one action is likely to be detrimental to performance in the long run and therefore it would become more important to mix up decisions.

5.2.6. System Training

How the system is trained is the final key area of the architecture that we discuss, in regard to system maintenance. One of the major benefits of producing case-based strategies via expert imitation, is that different types of strategies can be produced by simply modifying the data that is used to train the system. Decisions that were made by an expert player can be extracted from hand history logs and used to train a case-based strategy. Experts can be either human or other artificial agents.

In order to train a case-based strategy, perfect information is required, i.e. the data needs to record the hidden card information of the expert player. Typically, data collected from online poker sites only contains this information when the original expert played a hand that resulted in a *showdown*. For hands that were folded before a showdown, this information is lost. It is difficult to train a strategy on data where this information is missing. More importantly, any attempt to train a system on only the data where showdowns occurred would result in biased actions, as the decision to fold would never be encountered.

It is for these reasons that our case-based strategies have been trained on data made publicly available from the Annual Computer Poker Competition [1]. This data records hand history logs for all matches played between computerised agents at a particular year's competition. The data contains perfect information for every hand played and therefore can easily be used to train an imitation-based system. Furthermore, the computerised agents that participate at the ACPC each year are expected to improve in playing strength over the years and hence re-training the system on updated data should have a follow on affect on performance for any imitation strategies produced from the data. Our case-based strategies have typically selected subsets of data to train on, based on the decisions made by the agents that have performed the best in either of the two winner determination methods used by the ACPC.

There are both advantages and disadvantages for producing strategies that rely on generalising decisions from training data. While this provides a convenient mechanism for easily upgrading a system's play, there is an inherent reliance on the quality of the underlying data in order to produce reasonable strategies. Furthermore, it is reasonable to assume that strategies produced in this way are typically only expected to do as well as the original expert(s) they are trained on.

5.3. A Framework for Producing Case-Based Strategies in Two-Player, Limit Texas Hold'em

For the six key areas of our architecture (described above) maintenance was guided via com-

Table 4
A case is made up of three attribute-value pairs, which describe the current state of the game. A solution consists of an action and outcome triple, which records the average numerical value of applying the action (-∞ refers to an unknown outcome).

Attribute	Type	Example
1. Hand Strength	Integer	1 – 50
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, -∞, -∞), ...

parative evaluation and overall impact on performance. The outcome of this intensive, systematic maintenance is the establishment of a final framework for producing case-based strategies in the domain of two-player, limit Hold'em.

Here we present the details of the final framework we have established for producing case-based strategies. The following sections illustrate the details of our framework by specifying the following sufficient components:

1. A representation for encoding cases and game state information
2. The corresponding similarity metrics required for decision generalisation.

5.3.1. Case Representation

Table 4 depicts the final post-flop case representation used to capture game state information. A single case is represented by a collection of attribute-value pairs. Separate case-bases are constructed for the separate rounds of play by processing a collection of hand histories and recording values for each of the three attributes listed in Table 4. The attributes have been selected by the authors as they capture all the necessary information required to make a betting decision. Each of the post-flop attribute-value pairs are now described in more detail:

1. Hand Strength: The quality of a player's hand is represented in our framework by calculating the $E[HS^2]$ of the player's cards and then mapping these values into 1 out of 50 evenly divided buckets, i.e. uniform bucketing.

2. Betting Sequence: The betting sequence is represented as a string. It records all observed actions that have taken place in the current round, as well as previous rounds. Characters in the string are selected from the set of allowable actions, $A_{limit} = \{f, c, r\}$, rounds are delimited by a hyphen.

3. Board Texture: The board texture refers to important information available, given the combination of the publicly available community cards. In total, nine board texture categories were selected by the authors. These categories are displayed in Table 5 and are believed to represent salient information that any human player would notice. Specifically, the categories focus on whether it is possible that an opponent has made a flush (five cards of the same suit) or a straight (five cards of sequential rank), or a combination of both. The categories are broken up into **possible** and **highly-possible** distinctions. A category labelled **possible** refers to the situation where the opponent requires two of their personal cards in order to make their flush or straight. On the other hand, a **highly-possible** category only requires the opponent to use one of their personal cards to make their hand, making it more likely they have a straight or flush.

5.3.2. Similarity Metrics

Each feature requires a corresponding local similarity metric in order to generalise decisions contained in a set of data. Here we present the metrics specified by our framework.

1. Hand Strength: Equation 2 specifies the metric used to determine similarity between two hand strength buckets (f_1, f_2).

$$sim(f_1, f_2) = max\{1 - k \cdot \frac{|f_1 - f_2|}{T}, 0\} \quad (2)$$

Here, T refers to the total number of buckets that have been defined, where $f_1, f_2 \in [1, T]$ and k is a scalar parameter used to adjust the rate at which similarity should decrease.

2. Betting Sequence: To determine similarity between two betting sequences we developed a custom similarity metric that involves the identification of *stepped* levels of similarity, based on the number of bets/raises made by each player. 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.

3. Board Texture: To determine similarity between board texture categories a similarity matrix was derived. Matrix rows and columns in Fig. 3. represent the different categories defined in Table 5. Diagonal entries refer to two sets of community cards that map to the same category, in which case similarity is always 1. Non-diagonal entries refer to similarity values between two dissimilar categories. These values were hand picked by the authors. The matrix given in Fig. 3. is symmetric.

	A	B	C	D	E	F	G	H	I
A	1	0	0	0	0	0	0	0	0
B	0	1	0.8	0.7	0	0	0	0	0
C	0	0.8	1	0.7	0	0	0	0	0
D	0	0.7	0.7	1	0	0	0	0	0
E	0	0	0	0	1	0.8	0.7	0	0.6
F	0	0	0	0	0.8	1	0.7	0	0.5
G	0	0	0	0	0.7	0.7	1	0.8	0.8
H	0	0	0	0	0	0	0.8	1	0.8
I	0	0	0	0	0.6	0.5	0.8	0.8	1

Fig. 3. Board texture similarity matrix.

Table 5
Board Texture Key

A	No salient
B	Flush possible
C	Straight possible
D	Flush possible, straight possible
E	Straight highly possible
F	Flush possible, straight highly possible
G	Flush highly possible
H	Flush highly possible, straight possible
I	Flush highly possible, straight highly possible

5.4. Experimental Results

We now present a series of experimental results collected in the domain of two-player, limit Texas Hold'em. The results presented are obtained from annual computer poker competitions and data collected by challenging human opposition. For each evaluated case-based strategy, we provide an *architecture snapshot* that captures the relevant details of the parameters used for each of the six key architecture areas, that were previously discussed.

5.4.1. 2009 IJCAI Computer Poker Competition

We begin with the results of the 2009 ACPC, held at the International Joint Conference on Artificial Intelligence. Here, we submitted our case-based agent, Sartre, for the first time, to challenge other computerised agents submitted from all over the world. The following *architecture snapshot* depicts the details of the submitted agent:

1. Feature Representation
   (a) Hand Strength – *categories*
   (b) Betting Sequence – *string*
   (c) Board Texture – *categories*
2. Similarity Assessment – *all-or-nothing*
3. Solution Representation – *single*
4. Case Retrieval – *variable k*
5. Re-Use Policy – *max-frequency*
6. System Training – *Hyperborean-08*

The *architecture snapshot* above represents a baseline strategy where maintenance had yet to be performed. Each of the entries listed above corresponds to one of the six key architecture areas introduced in Section 5.2. Notice that trivial *all-or-nothing* similarity was employed along with a *single action solution representation*, which resulted in a redundant case-base. The value for *system training* refers to the original expert whose decisions were used to train the system.

The final results are displayed in Table 6. The competition consisted of two winner determination methods: bankroll *instant run-off* and *total bankroll*. Each agent played between 75 and 120 duplicate matches against every other agent in order to obtain the average values displayed. Each match consisted of 3000 duplicate hands. The values presented are the number of small bets per hand won or lost. Our case-based agent, Sartre, achieved a 7^{th} place finish in the instant run-off division and a 6^{th} place finish in the total bankroll division.

5.4.2. 2010 AAAI Computer Poker Competition

Following the maintenance experiments presented in Section 5.2, an updated case-based strategy was submitted to the 2010 ACPC, held at the Twenty-Forth AAAI Conference on Artificial Intelligence. Our entry, once again named Sartre, used the following *architecture snapshot*:

1. Feature Representation
   (a) Hand Strength – *50 buckets E[HS^2]*
   (b) Betting Sequence – *string*
   (c) Board Texture – *categories*
2. Similarity Assessment
   (a) Hand Strength – *Euclidean*
   (b) Betting Sequence – *custom*
   (c) Board Texture – *matrix*
3. Solution Representation – *vector*
4. Case Retrieval – *k = 1*
5. Re-Use Policy – *probabilistic*
6. System Training *MANZANA*

Here a vector valued *solution representation* was used together with improved similarity assessment. Given the updated solution representation, a single nearest neighbour, $k = 1$, was retrieved via the k-NN algorithm. A *probabilistic* solution re-use policy was employed and the system was trained on the decisions of the winner of the 2009 total bankroll division. The final results are presented in Table 7. Once again two winner determination divisions are presented and the values are depicted in small bets per hand with 95% confidence intervals. Given the improvements, Sartre was able to achieve a 6^{th} place finish in the runoff division and a 3^{rd} place finish in the total bankroll division.

5.4.3. 2011 AAAI Computer Poker Competition

The 2011 ACPC was held at the Twenty-Fifth AAAI Conference on Artificial Intelligence. Our entry to the competition is represented by the following *architecture snapshot*:

1. Feature Representation
   (a) Hand Strength – *50 buckets E[HS^2]*
   (b) Betting Sequence – *string*
   (c) Board Texture – *categories*
2. Similarity Assessment
   (a) Hand Strength – *Euclidean*
   (b) Betting Sequence – *custom*
   (c) Board Texture – *matrix*

Table 6
2009 limit heads up bankroll and runoff results. Values are in sb/h with 95% confidence intervals.

Limit Heads up Bankroll Runoff	Limit Heads up Total Bankroll	Average Won/Lost
1. GGValuta	1. MANZANA	0.186 ± 0.002
2. Hyperborean-Eqm	2. Hyperborean-BR	0.116 ± 0.002
3. MANZANA	3. GGValuta	0.110 ± 0.002
4. Rockhopper	4. Hyperborean-Eqm	0.116 ± 0.002
5. Hyperborean-BR	5. Rockhopper	0.103 ± 0.002
6. Slumbot	**6. Sartre**	0.097 ± 0.002
7. Sartre	7. Slumbot	0.096 ± 0.002
8. GS5	8. GS5	0.082 ± 0.002
9. AoBot	9. AoBot	-0.002 ± 0.003
10. GS5Dynamic	10. dcurbHU	-0.07 ± 0.002
11. LIDIA	11. LIDIA	-0.094 ± 0.002
12. dcurbHU	12. GS5Dynamic	-0.201 ± 0.002
13. Tommybot		

Table 7
2010 limit heads up bankroll and runoff results. Values are in sb/h with 95% confidence intervals.

Limit Heads up Bankroll Runoff	Limit Heads up Total Bankroll	Average Won/Lost
1. Rockhopper	1. PULPO	0.225 ± 0.003
2. GGValuta	2. Hyperborean-TBR	0.207 ± 0.002
3. Hyperborean-IRO	**3. Sartre**	0.203 ± 0.002
4. Slumbot	4. Rockhopper	0.200 ± 0.002
5. PULPO	5. Slumbot	0.199 ± 0.002
6. Sartre	6. GGValuta	0.193 ± 0.003
7. GS6-IRO	7. Jester	0.164 ± 0.003
8. Arnold2	8. Arnold2	0.160 ± 0.003
9. Jester	9. GS6-TBR	0.139 ± 0.004
10. LittleRock	10. LittleRock	0.118 ± 0.003
11. PLICAS	11. PLICAS	-0.046 ± 0.005
12. ASVP	12. ASVP	-0.320 ± 0.006
13. longhorn	13. longhorn	-1.441 ± 0.005

3. Solution Representation – *vector*
4. Case Retrieval – $k = 1$
5. Re-Use Policy – *max-frequency*
6. System Training *combination*

While reasonably similar to the strategy employed for the 2010 competition, the *architecture snapshot* above exhibits some important differences. In particular, the 2011 agent consisted of a combination of multiple strategies that were each trained by observing and generalising the decisions of separate original experts, see [29] for further details. Also notice the switch from a *probabilistic* solution re-use policy to a *max-frequency* policy.

Table 8 presents results from the 2011 competition. For the third year in a row, Sartre was able to improve its performance in both winner determination divisions. Sartre improved from 6^{th} to 4^{th} place in the *instant runoff* division. Whereas, in the *total bankroll* division Sartre improved from 3^{rd} place to 2^{nd} place. Notice however, while Calamari was declared the winner of this competition (with Sartre placing 2nd), there is overlap in the values used to make this decision, given the standard deviations.

5.4.4. Human Opponents – Limit

Finally, we provide results for our *case-based strategies* when challenging human opposition. These results were collected from an online web application[5] where any human opponent was able

[5] http://www.cs.auckland.ac.nz/poker/

Table 8
2011 limit heads up bankroll and runoff results. Values are in sb/h with 95% confidence intervals.

Limit Heads up Bankroll Runoff	Limit Heads up Total Bankroll	Average Won/Lost
1. Hyperborean-2011-2p-limit-iro	1. Calamari	0.286 ± 0.004
2. Slumbot	2. **Sartre**	0.281 ± 0.006
3. Calamari	3. Hyperborean-2011-2p-limit-tbr	0.225 ± 0.005
4. **Sartre**	4. Feste	0.220 ± 0.005
5. LittleRock	5. Slumbot	0.216 ± 0.006
6. ZBot	6. ZBot	0.209 ± 0.006
7. GGValuta	7. Patience	0.209 ± 0.006
8. Feste	8. 2Bot	0.192 ± 0.006
9. Patience	9. LittleRock	0.180 ± 0.005
10. 2Bot	10. GGValuta	0.145 ± 0.006
11. RobotBot	11. AAIMontybot	0.001 ± 0.012
12. AAIMontybot	12. RobotBot	-0.035 ± 0.010
13. Entropy	13. GBR	-0.051 ± 0.012
14. GBR	14. Entropy	-0.097 ± 0.013
15. player.zeta	15. player.zeta	-0.176 ± 0.017
16. Calvin	16. Calvin	-0.230 ± 0.012
17. Tiltnet.Adaptive	17. Tiltnet	-0.268 ± 0.010
18. POMPEIA	18. POMPEIA	-0.549 ± 0.006
19. TellBot	19. TellBot	-0.759 ± 0.016

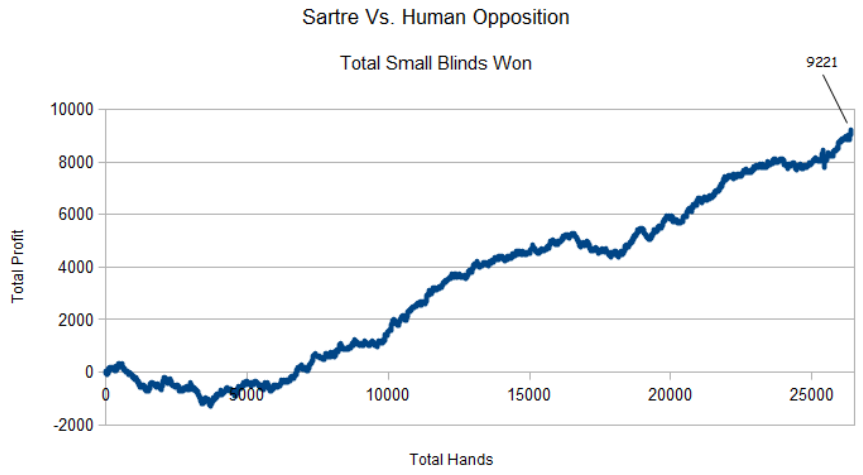

Fig. 4. The number of small bets won in total against all human opponents to challenge Sartre.

to log on via their browser and challenge the latest version of the Sartre system. While it is interesting to gauge how well Sartre performs against human opponents (not just computerised agents), care must also be taken in interpreting these results as there has been no effort made to restrict the human opposition to only player's of a certain quality.

Fig. 4. depicts the number of small bets won in total against every human opponent to challenge the system. In total just under 30,000 poker hands have been recorded and Sartre currently records a profit of 9221 small bets. This results in a final small bets per hand (sb/h) value of 0.30734. Fig. 5. depicts the sb/h values recorded over all hands.

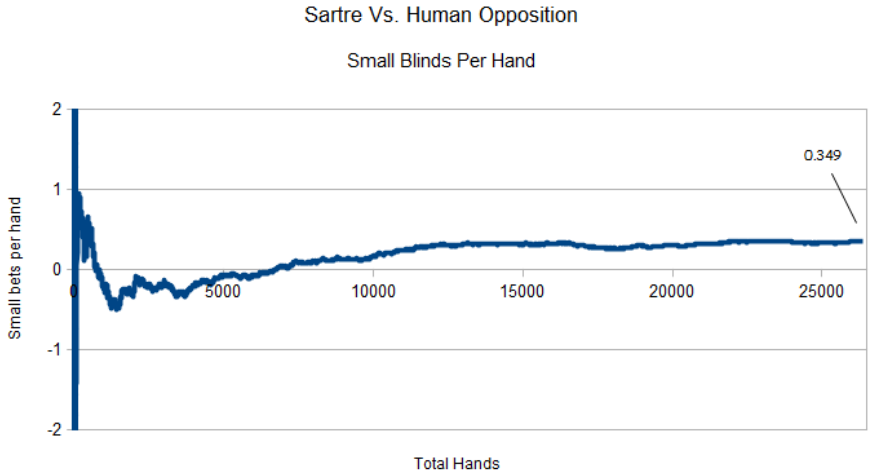

Fig. 5. The small bets per hand (sb/h) won by Sartre over every hand played.

5.4.5. Discussion

The results show a marked improvement between outcomes obtained over the years at the annual computer poker competition. In particular, Sartre achieves a 6^{th} place finish in the total bankroll division at the 2009 competition. The following year, using the updated strategy, Sartre now achieves a 3^{rd} place finish (out of a total of 13 agents) in the same event. Finally, Sartre was declared *runner-up* at the 2011 *total bankroll* division and very nearly wins this competition. These results suggest that the maintenance performed and the updated architecture does indeed have a significant impact on the quality of the case-based strategies produced. Furthermore, it is expected that those agents that competed in the previous year's competition have improved between competitions as well.

For the data collected against human opposition, while we can not comment on the quality of the human opponents challenged, the curves in Figs. 4. and 5. show a general upward trend and a steady average profit, respectively.

6. Two-Player, No-Limit Texas Hold'em

We now examine the application of case-based strategies in the more complicated domain of two-player, no-limit Texas Hold'em. Here we take into consideration the lessons learned during the maintenance that was performed in the two-player, limit Hold'em domain. We use this information and the insights obtained in the limit domain to establish a finalised framework in the no-limit domain. However, before no-limit case-based strategies can be produced, the difficulties of handling a no-limit betting structure need to be addressed.

In the no-limit variation players' bet sizes are no longer restricted to fixed amounts, instead a player can wager any amount they wish, up to the total amount of chips they possess. This simple rule change has a profound effect on the nature of the game, as well as on the development of computerised agents that wish to handle a no-limit betting structure. In particular, the transition to a no-limit domain results in unique challenges that are not encountered in a limit poker environment. First, there is the issue of establishing a set of abstract betting actions that all real actions will be mapped into during game play. This is referred to as **action abstraction** and it allows the vast, continuous domain of no-limit Hold'em to be approximated by a much smaller *abstract* state space. Second, given an established set of abstract actions, a **translation** process is required that determines how *best* to map real actions into their appropriate abstract counterparts, as well as a reverse translation that maps abstract actions back into appropriate real-world betting decisions.

6.1. Abstraction

Abstraction is a concept used by game theoretic poker agents that derive ϵ-Nash equilibrium strategies for the game of Texas Hold'em. As the actual Hold'em game tree is much too large to represent and solve explicitly, it becomes necessary to impose certain abstractions that help restrict the size of the original game. For Texas Hold'em, there are two main types of abstraction:

1. **Chance abstraction** – which reduces the number of chance events that are required to be dealt with. This is typically achieved by grouping strategically similar hands into a restricted set of buckets.
2. **Action abstraction** – which restricts the number of actions a player is allowed to perform.

Action abstractions can typically be avoided by poker agents that specialise in limit poker, where there are only 3 actions to choose from: fold (f), check/call (c) or bet/raise (r). However in no-limit, where a raise can take on any value, some sort of action abstraction is required. This is achieved by restricting the available bet/raise options to a discrete set of categories based on fractions of the current pot size. For example, a typical abstraction such as: *fcpa*, restricts the allowed actions to:

- f – fold,
- c – call,
- p – bet the size of the pot
- a – all-in (i.e. the player bets all their remaining chips)

Given this abstraction, all actions are interpreted by assigning the actual actions into one of their abstract counterparts. While our case-based strategies do not attempt to derive an ϵ-Nash equilibrium solution for no-limit Hold'em, they are still required to define an action abstraction in order to restrict the number of actions allowed in the game and hence reduce the state space.

6.2. Translation

Given that all bets need to be mapped into one of the abstract actions, a translation process is required to define the appropriate mapping. The choice of translation needs to be considered carefully as some mappings can be easily exploited.

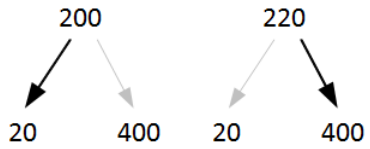

Fig. 6. Using the *fcpa* abstraction, the amounts above will either be mapped as a *pot* sized bet – 20 or an *all-in* bet – 400.

The following example illustrates how the choice of translation can lead to exploitability.

Consider a translation that maps bets into abstract actions based on absolute distance, i.e. the abstract action that is closest to the bet amount is the one that the bet gets mapped in to. Given a pot size of 20 chips and the *fcpa* abstraction (from above) any bet between 20 and a maximum of 400 chips will either be mapped into a *pot* (p) sized bet or an *all-in* (a) bet. Using this translation method, a bet amount of 200 chips will be considered a pot-sized bet, whereas a bet amount of only 20 chips more, 220, will be considered an all-in bet. See Fig. 6.

There can be various benefits for a player in making their opponent think that they have made a pot-sized bet or an all-in bet. First, consider the situation where an *exploitive* player bets 220 chips. This bet amount will be mapped into the *all-in* abstract action by an opponent that uses the *fcpa* abstraction. In other words, a player has made their opponent *believe* that they have bet all 400 of their chips, when in reality they have only risked 220. In this situation, it is likely that an opponent will fold most hands to an *all-in* bet, however even when the opponent calls, the *exploitive* player has still only wagered 220 chips as opposed to 400.

On the other hand, by betting just 20 chips less (i.e. 200 instead of 220), this alternative situation can have an even more dramatic effect on the outcome of a hand. When an *exploitive* player makes a bet of 200 chips this bet amount will be mapped as a pot-sized bet and if their opponent decides to call they will *believe* that they have only contributed 20 chips to the pot when in reality they would have invested 200 chips. If this is followed up with a large *all-in* bet, an opponent that believes they have only invested 20 chips in the pot is much more likely to fold a mediocre hand than a player that has contributed ten times that amount. As such, an *exploitive* player has the ability to make

their opponent believe they are only losing a small proportion of their stack size by folding, when in reality they are losing a lot more. This can lead to large profits for the *exploitive* player.

The above example shows that a translation method that uses deterministic mapping based on absolute distance has the ability to be exploited simply by selecting particular bet amounts. Schnizlein *et al.* [30] formalise this type of translation as *hard translation*. **Hard translation** is a many to one mapping that maps an unabstracted betting value into an abstract action based on a chosen distance metric. Given a unique unabstracted betting value, hard translation will always map this value into the same abstract action. A disadvantage of hard translation is that an opponent can exploit this mapping simply by selecting particular betting values. To overcome this problem, a more robust translation procedure is required, one that cannot be exploited in such a way. Schnizlein *et al.* [30] also formalise an alternative *soft translation* that addresses some of the shortcomings of hard translation. **Soft translation** is a probabilistic state translation that uses normalised weights as similarity measures to map an unabstracted betting value into an abstract action. The use of a probabilistic mapping ensures that soft translation cannot be exploited like hard translation can.

Having considered the issues to do with *abstraction* and *translation*, we are now able to present our framework for producing case-based strategies in the domain of no-limit Hold'em.

6.3. A Framework for Producing Case-Based Strategies in Two-Player, No-Limit Texas Hold'em

We now present the final framework we have established for producing case-based strategies in the domain of two-player, no-limit Texas Hold'em. In order to define the framework it is necessary to specify the following four conditions:

1. The action abstraction used
2. The type of state translation used and where this occurs within the architecture
3. A representation for encoding cases and game state information
4. The corresponding similarity metrics required for decision generalisation.

Table 9
The action abstraction used by our case-based strategies.

f	fold
c	call
q	quarter pot
h	half pot
i	three quarter pot
p	pot
d	double pot
v	five times pot
t	ten times pot
a	all in

The conditions specified above are an extrapolation of those that were used to define the framework for producing case-based strategies in the limit Hold'em domain.

6.3.1. Action Abstraction

Within our framework, we use the following action abstraction: *fcqhipdvta*. Table 9 provides an explanation of the symbols used.

6.3.2. State Translation

Define $A = \{q, h, i, p, d, v, t, a\}$ to be the set of abstract betting actions. Actions f and c are omitted from A as these require no mapping. The exact translation parameters that are used differ depending on where translation takes place within the system architecture, as follows:

1. During case-base construction hand history logs from previously played hands are required to be encoded into cases. Here *hard translation* is specified by the following function $T_h : \Re \rightarrow A$:

$$T_h(b) = \begin{cases} x & \text{if } \frac{x}{b} > \frac{b}{y} \\ y & \text{otherwise} \end{cases} \quad (3)$$

where $b \in \Re$ is the proportion of the total pot that has been bet in the actual game and $x, y \in A$ are abstract actions that map to actual pot proportions in the real game and $x <= b < y$. The fact that hard translation has the capability to be exploited is not a concern during case-base construction. Hard translation is used during this stage to ensure that re-training the system with the same hand history data will result in the same case-base.

2. During actual game play real betting actions observed during a hand are required to be mapped into appropriate abstract actions. This is equivalent to the translation process required of ϵ-Nash equilibrium agents that solve an abstract extensive form game, such as [5,15,30]. Observant opponents have the capability to exploit deterministic mappings during game play, hence a soft translation function is used for this stage, $T_s : \Re \rightarrow A$, given by the following probabilistic equations:

$$P(x) = \frac{\frac{x}{b} - \frac{x}{y}}{1 - \frac{x}{y}} \quad (4)$$

$$P(y) = \frac{\frac{b}{y} - \frac{x}{y}}{1 - \frac{x}{y}} \quad (5)$$

where once again, $b \in \Re$ is the proportion of the total pot that has been bet in the actual game and $x, y \in A$ are abstract actions that map to actual pot proportions in the real game and $x <= b < y$. Note that when $b = x, P(x) = 1$ and $P(y) = 0$ and when $b = y, P(x) = 0$ and $P(y) = 1$. Hence, a betting action that maps directly to an abstract action in A does not need to be probabilistically selected. On the other hand, when $b \neq x$ and $b \neq y$, abstract actions are chosen probabilistically. Note that in Equations (4) and (5), $P(x) + P(y) \neq 1$ and hence a final abstract action is probabilistically chosen by first normalising these values.

3. A final reverse translation phase is required to map a chosen abstract action into a real value to be used during game play. A *reverse mapping* is required to map the abstract action into an appropriate real betting value, given the current game conditions. The following function is used to perform reverse translation, $T_r : A \rightarrow \Re$:

$$T_r(x) = x' \pm \Delta x' \quad (6)$$

where $x \in A$ and $x' \in \Re$ is the real value corresponding to abstract action x and $\Delta x'$ is some random proportion of the bet amount that is used to ensure the strategy does not always map abstract actions to their exact real world counterparts. Randomisation is used to limit any exploitability that could be introduced by consistently betting the same amount. For example, when $x' = 100$ and $\Delta = 0.3$, any amount between 70 and 130 chips may be bet.

6.3.3. Case Representation

Table 10 depicts the case representation used to capture important game state information in the domain of two-player, no-limit Texas Hold'em. As in the limit domain, a case is represented by a collection of attribute-value pairs and separate case-bases are constructed for the separate betting rounds by processing a collection of hand histories and recording values for each of the attributes listed.

Three of the four attributes (*hand strength*, *betting sequence*, *board texture*) are the same as those used within the limit framework. The *stack commitment* attribute was introduced especially for the no-limit variation of the game. All attributes were selected by the authors, given their importance in determining a final betting decision.

Each case also records a solution. Once again a solution is made up of an action vector and an outcome vector. The entries within each vector correspond to a particular betting decision. Given the extended set of betting actions that are available in the no-limit domain, the solution vectors are represented as n-tuples (as opposed to triples, which were used in the limit domain). Once again, the entries within the action vector must sum to one.

Each of the attribute-value pairs are described in more detail below.

1. **Hand Strength:** As in the limit domain, a player's hand is represented by calculating the E[HS^2] of the player's cards and mapping these values into 1 out of 50 possible buckets. A standard bucketing approach is used where the 50 possible buckets are evenly divided.
2. **Betting Sequence:** Once again a string is used to represent the *betting sequence*, which records all observed actions that have taken place in the current round, as well as previous rounds. Notice however, that the characters used to represent the betting sequence can be any of the abstract actions defined in Table 9. As such, there are a lot more possible no-limit betting sequences than there are limit sequences. Once again rounds are delimited by hyphens.

Table 10
The case representation used for producing case-based strategies in the domain of no-limit Texas Hold'em.

Feature	Type	Example
1. Hand Strength	Integer	1 – 50
2. Betting Sequence	String	*pdc-cqc-c*, *cc-*, *dc-qc-ci*, ...
3. Stack Commitment	Integer	1,2,3,4
4. Board Texture	Class	*No-Salient*, *Flush-Possible*, *Straight-Possible*, *Flush-Highly-Possible*, ...
Action	n-tuple	(0.0, 1.0, 0.0, 0.0, ...), ...
Outcome	n-tuple	(-∞, 36.0, -∞, -∞, ...), ...

3. Stack Commitment: In the no-limit variation of Texas Hold'em players can wager any amount up to their total stack size. The proportion of chips committed by a player, compared to the player's stack size, is therefore of much greater importance, compared to limit Hold'em. The betting sequence maps bet amounts into discrete categories based on their proportion of the pot size. This results in information that is lost about the total amount of chips a player has contributed to the pot, relative to the size of their starting stack. Once a player has contributed a large proportion of their stack to a pot, it becomes more important for that player to remain in the hand, rather than fold, i.e. they have become **pot committed**.
The **stack commitment** feature maps this value into one of N categories, where N is a specified granularity:

$$[0-\frac{1}{N}], [\frac{1}{N}-\frac{2}{N}], \ldots, [\frac{N-2}{N}-\frac{N-1}{N}][\frac{N-1}{N}-1]$$

Hence, for a granularity of $N = 4$, a stack commitment of 1 means the player has committed less than 25% of their initial stack, a stack commitment of 2 means that player has contributed somewhere between 25% and 50% of their total stack, and so forth.

4. Board Texture: The details of the board texture attribute are exactly as those described in the limit domain (see Section 5.3.1).

6.3.4. Similarity Metrics

Each feature requires a corresponding local similarity metric in order to generalise decisions contained in a set of data. Here we present the metrics specified by our framework.

1. Hand Strength: Similarity between hand strength values is determined by the same metric used in the limit Hold'em domain, as specified by Equation 2.

2. Betting Sequence: Recall that the following bet discretisation is used: *fcqhipdvta*. Within this representation there are some non-identical bet sizes that are reasonably similar to each other. For example, a bet of half the pot (h) is quite close to a bet of three quarters of the pot (i). The betting sequence similarity metric we derived compares bet sizes against each other that occur at the same location within two betting sequences.
Let S_1 and S_2 be two betting sequences made up of actions $a \in A \cup \{f, c\}$, where the notation $S_{1,i}, S_{2,i}$ refers to the i^{th} character in the betting sequences S_1 and S_2, respectively.
For two betting sequences to be considered similar they first need to satisfy the following conditions:

1. $|S_1| = |S_2|$
2. $S_{1,i} = c \Rightarrow S_{2,i} = c$, and
   $S_{1,j} = a \Rightarrow S_{2,j} = a$

i.e. each sequence contains the same number of elements and any calls (c) or all-in bets (a)

Table 11
Bet Discretisation String

q	h	i	p	d	v	t

that occur within sequence S_1 must also occur at the same location in sequence S_2[6].

Any two betting sequences that do not satisfy the initial two conditions above are assigned a similarity value of 0. On the other hand, if the two betting sequences do satisfy the above conditions their bet sizes can then be compared against each other and a similarity value assigned.

Exactly how dissimilar two individual bets are to each other can be quantified by how far away from each other they occur within the bet discretisation string, displayed in Table 11.

As h and i are neighbours in the discretisation string they can be considered to occur at a distance of 1 away from each other, $\delta(h, i) = 1$, as opposed to say $\delta(q, t) = 6$, which are at opposite ends on the discretisation string.

For two betting sequences S_1, S_2 overall similarity is determined by (7):

$$sim(S_1, S_2) = \begin{cases} 1 - \sum_{i=0}^{|S_1|} \delta(S_{1,i}, S_{2,i})\alpha & \text{if } |S_1| = |S_2|, \\ & S_{1,i} = c \Rightarrow S_{2,i} = c, \\ & S_{1,j} = a \Rightarrow S_{2,j} = a \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where α is some constant rate of decay.

The following is a concrete example of how similarity is computed for two non-identical betting sequences. Consider two betting sequences, $S_1 = ihpc$ and $S_2 = dqpc$. Here, $|S_1| = 4$ and $|S_2| = 4$ and wherever there exists a check/call (c) in S_1, there exists a corresponding c in S_2. As both conditions are satisfied we can evaluate the top half of Equation (7):

$$\begin{aligned} sim(S_1, S_2) &= 1 - [\delta(i, d)\alpha + \delta(h, q)\alpha + \delta(p, p)\alpha + \delta(c, c)\alpha] \\ &= 1 - [2 \cdot \alpha + 1 \cdot \alpha + 0 \cdot \alpha + 0 \cdot \alpha] \\ &= 1 - 3\alpha \end{aligned}$$

Using a rate of decay of $\alpha = 0.05$, gives a final similarity of: $1 - 0.15 = 0.85$.

3. **Stack Commitment:** The stack commitment metric uses an exponentially decreasing function.

$$sim(f_1, f_2) = e^{(-|f_1 - f_2|)} \tag{8}$$

where, $f_1, f_2 \in [1, N]$ and N refers to the granularity used for the stack commitment attribute. This function was chosen as small differences between two stack commitment attributes (f_1, f_2) should result in large drops in similarity.

4. **Board Texture:** Similarity between board texture values is determined by the same similarity matrix used in the limit Hold'em domain, as specified in Fig. 3.

6.4. Experimental Results

Once again we provide experimental results that have been obtained from annual computer poker competitions, where our case-based strategies have challenged other computerised agents. We also provide results against human opposition.

6.4.1. 2010 AAAI Computer Poker Competition

We submitted an early version of our two-player, no-limit case-based strategy to the 2010 computer poker competition. The submitted strategy was trained on the hand history information of the previous year's winning agent (i.e Hyperborean) and used a probabilistic decision re-use policy.

In the no-limit competition, the *Doyle's Game* rule variation is used where both players begin each hand with 200 big blinds. All matches played were duplicate matches, where each competitor played 200 duplicate matches (each consisting of $N = 3000$ hands) against every other competitor. The final results are displayed in Table 12. Our entry is SartreNL.

Out of a total of five competitors, SartreNL placed 4^{th} in the *total bankroll* division and 2^{nd} in the *instant runoff* division. One important thing to note is that the strategy played by SartreNL was an early version and did not fully adhere to the framework described in Section 6.3. In particular, the following abstraction was used: *fcqhipdvt*. While this abstraction is very similar to that described in Section 6.3, it is missing the *all-in* action, as this was only added after the agent had been submitted. A further difference is that the strategy submitted only used hard translation.

[6] A betting sequence consists of one or more betting rounds, the above conditions must be satisfied for all betting rounds within the betting sequence.

Table 12
2010 no-limit heads up bankroll and runoff results. Values are in big blinds per hand with 95% confidence intervals.

No-Limit Heads up Bankroll Runoff	No-Limit Heads up Total Bankroll	Average Won/Lost
1. Hyperborean.iro	1. Tartanian4.tbr	2.156 ± 0.048
2. **SartreNL**	2. PokerBotSLO	1.458 ± 0.184
3. Tartanian4.iro	3. Hyperborean.tbr	1.212 ± 0.026
4. PokerBotSLO	4. **SartreNL**	0.537 ± 0.034
5. c4tw.iro	5. c4tw.tbr	-5.362 ± 0.201

Table 13
2011 no-limit heads up bankroll and runoff results. Values are in big blinds per hand with 95% confidence intervals.

No-Limit Heads up Bankroll Runoff	No-Limit Heads up Total Bankroll	Average Won/Lost
1. Hyperborean-2011-2p-nolimit-iro	1. Lucky7	1.567 ± 0.053
2. **SartreNL**	2. **SartreNL**	1.302 ± 0.042
3. hugh	3. Hyperborean-2011-2p-nolimit-tbr	1.133 ± 0.026
4. Rembrant	4. player.kappa.nl	1.026 ± 0.105
5. Lucky7	5. hugh	0.968 ± 0.054
6. player.kappa.nl	6. Rembrant	0.464 ± 0.024
7. POMPEIA	7. POMPEIA	-6.460 ± 0.051

6.4.2. 2011 AAAI Computer Poker Competition

The agent submitted to the 2011 competition fully adhered to the framework described in Section 6.3. Our entry was trained on hand history information from the winner of the previous year's bankroll instant run-off competition (once again Hyperborean). A *max-frequency* solution re-use policy was employed. In the 2011 competition, SartreNL placed 2^{nd} in both winner determination divisions, out of a total of seven competitors. Table 13 presents the final results.

6.4.3. Human Opponents – No-Limit

Once again, we have gathered results against human opponents in the no-limit domain via our browser based application. As with the real world limit Hold'em results (see Section 5.4.4), these results are presented as a guide only and care must be taking in drawing any final conclusions. Fig. 7. records the number of big blinds won in total against all human opponents. SartreNL achieves a final profit of 7539.5 big blinds in just under 9000 hands. Fig. 8. shows the big blinds per hand (bb/h) won over all hands played, the final bb/h value recorded is 0.9023.

6.4.4. Discussion

The results of the 2010 ACPC were somewhat mixed as SartreNL performed very well in the *instant runoff* division, placing 2^{nd}, but not so well in the *total bankroll* division where the average profit won $+0.537 \pm 0.034$ was lower than all but one of the competitors. This relatively poor overall profit is most likely due to the 2010 version of SartreNL not encoding an *all-in* action in its abstraction, which would have limited the amount of chips that could be won.

After updating the strategy for the 2011 competition to accurately reflect the framework described, SartreNL's performance and total profit improves to $+1.302 \pm 0.042$. As such SartreNL is able to perform well in both winner determination procedures, placing 2^{nd} in both divisions of the 2011 competition out of seven competitors.

While it is interesting to get an initial idea of how SartreNL does against human opponents, it would be unwise to draw any conclusions from this data, especially because not nearly enough hands have been played (given the variance involved in the no-limit variation of the game) to make any sort of accurate assessment.

7. Multi-Player, Limit Texas Hold'em

The final sub-domain that we have applied and evaluated case-based strategies within is multi-player Texas Hold'em. Specifically, three-player, limit Texas Hold'em, where an agent is required to challenge two opponents instead of just one.

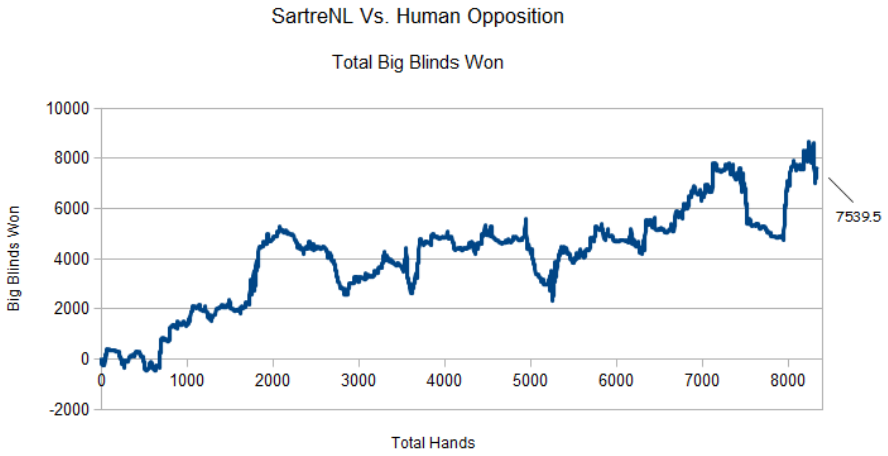

Fig. 7. The number of big blinds won in total against every human opponent to challenge SartreNL.

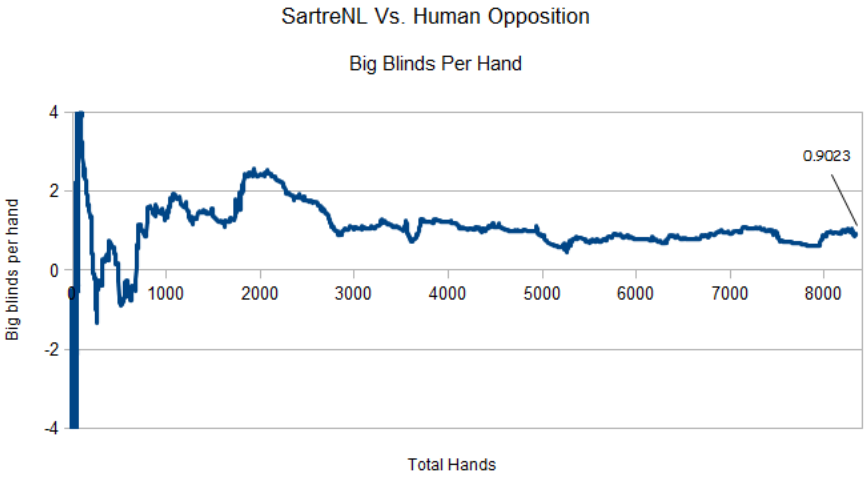

Fig. 8. The big blinds per hand (bb/h) won by SartreNL over every hand played.

Once again, we use the lessons learned by applying maintenance in the two-player, limit Hold'em domain in order to finalise a framework that can handle multi-player betting. An interesting question that arises within the three-player domain is whether we can make use of the case-based strategies that have already been developed in the two-player domain, and if so, how do we determine a suitable mapping between domains? Before presenting our final framework for constructing multi-player case-based strategies we investigate the efficacy of *strategy switching* between domains.

7.1. Strategy Switching

In the three-player domain, when one opponent *folds* it would be useful if the case-based strategies, previously developed for heads-up play, could be used to make a betting decision. A type of switching strategy was described in [27] for game theoretic multi-player poker agents that solved two-player sub-games for a small selection of initial betting sequences. Actions were chosen from an appropriate sub-game when one of these preselected sequences occurred in the real game. The *strategy switching* approach that we describe differs from

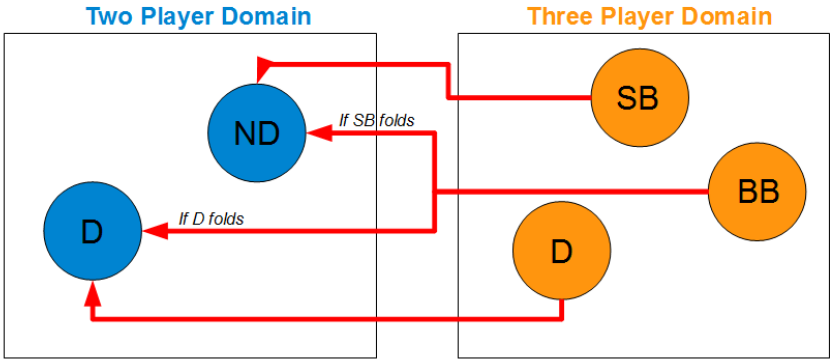

Fig. 9. Demonstrates how inter-domain mapping is performed for betting sequences when a fold takes place on the first betting round in the three player domain. Each node represents a particular player position within their domain.

that in [27] as our approach does not solve a selection of sub-games, but rather attempts to associate three-player betting sequences with appropriate two-player betting sequences, so that a two-player case-base can be searched instead. Consider the following pseudocode, where s refers to the choice of a particular strategy:

$s \leftarrow \emptyset$
if fold occurred **then**
  if $\exists$ inter-domain mapping **then**
    $s \leftarrow$ *heads-up* strategy
  else
    $s \leftarrow$ *multi-player* strategy
  end if
else
  $s \leftarrow$ *multi-player* strategy
end if

The pseudocode above requires a mapping between two separate domains in order to allow a heads-up strategy to be applied within a multi-player environment. One way that this can be achieved is to develop a similarity metric that is able to gauge how similar a game state encountered in a multiple-player domain is, compared to a corresponding state in heads-up play. Given our case representation for heads-up strategies, presented in Table 4, we require a suitable inter-domain mapping for the *betting sequence* attribute.

7.1.1. Inter-Domain Mapping

A mapping can occur between domains as long as one player has folded in the three-player domain. There are various ways an inter-domain mapping can take place between a three-player betting sequence and a two-player betting sequence. To determine how similar a three-player betting sequence is to a two-player betting sequence, we need to first map the actions of the two remaining players in the three-player domain to actions of an appropriate player in the two-player domain. An appropriate player is selected by considering the order in which the players act. We refer to this as an inter-domain *player mapping*.

We can determine how similar two betting sequences from separate domains are by counting the total number of *bet/raise* decisions taken by an active player in the three-player domain and comparing this with the total number of *bet/raise* decisions made by the corresponding player in the two player domain, as specified by the *player mapping*. This ensures that the mapping between domains retains the strength that each player exhibited by betting or raising.

Fig. 9. illustrates a possible *player mapping* that takes place when a *fold* has occurred during the preflop round of play. The abbreviations in Fig. 9. stand for *Dealer* (D), *Non Dealer* (ND), *Small Blind* (SB) and *Big Blind* (BB). The connections between the two domains (i.e. the red arrows) specifies the *player mapping* and are further explained below.

1. $D \rightarrow D$: As the *Dealer* is the last player to act postflop (in both domains) the total number of preflop raises made by D in the three-player domain, must match the total number of preflop raises made by D in the two-player domain for the inter-domain sequences to be considered similar.

2. $SB \rightarrow ND$: As the SB is the first player to act postflop (in the three-player domain) and the ND is the first player to act postflop (in the two-player domain), the total number of preflop raises made by the SB must match the total number of preflop raises made by the ND for the inter-domain sequences to be considered similar.
3. $BB \rightarrow D/ND$: For the final mapping, the order in which the BB acts depends on which player folded in the three-player domain (SB or D). If the SB folded, the BB will act first postflop and hence, the total number of preflop raises made by the BB must match the total number of preflop raises made by the ND. On the other hand, if the D was the player that folded, the BB will act last postflop and hence, the total number of preflop raises made by the BB must match the total number of preflop raises made by the D instead.

The above mapping works by ensuring a two-player betting sequence exists where the total number of preflop raise actions made by each player are equal to the total preflop raise actions made by the two remaining players in the three-player domain. Results obtained with the above mapping are presented in Section 7.3.1.

7.2. A Framework for Producing Case-Based Strategies in Multi-Player, Limit Texas Hold'em

We now present the final framework we have established for producing multi-player, limit case-based strategies, with or without *strategy switching*. Our framework consists of:

1. A representation for encoding cases and game state information
2. The corresponding similarity metrics required for decision generalisation.
3. When *strategy switching* is enabled, separate similarity metrics for inter-domain mapping.

7.2.1. Case Representation

Table 14 depicts the final post-flop case representation used to capture game state information in the multi-player, limit Hold'em domain. Notice that the representation used for multi-player cases is almost identical to that presented in Table 4. This similarity between representations is beneficial as it simplifies the inter-domain mapping that we require for *strategy switching*. The main difference in Table 14 has to do with the values used to represent the *betting sequence* attribute. As multi-player sequences involve more players, the sequences that record betting actions are larger and more complicated than those in the two-player domain.

Each of the attribute-value pairs are described in more detail below.

1. Hand Strength: As in the previous domains, a player's hand is represented by calculating the E[HS^2] of the player's cards and mapping these values into 1 out of 50 possible buckets via standard bucketing.
2. Betting Sequence: The details of the betting sequence attribute are similar to those described in the two-player, limit domain where actions are chosen from $A_{limit} = \{f, c, r\}$, and rounds are delimited by hyphens. However, as more players are involved in a hand, the betting sequences are different than those produced in the two-player domain.
3. Board Texture: The nine board texture categories, defined in Table 5, are re-used in the multi-player domain to capture important public card information.

7.2.2. Similarity Metrics

Once again, each feature requires a corresponding local similarity metric in order to generalise decisions contained in a set of data.

1. Hand Strength: Similarity between hand strength values is determined by the same metric used in the two-player limit Hold'em domain, as specified by Equation 2.
2. Betting Sequence: The metric used to determine similarity between *betting sequences* differs depending on whether *strategy switching* is enabled or not.

Without Switching When strategy switching is not required a three-player case-base will be searched that contains three-player betting sequences only. To determine similarity between these sequences, a stepped level similarity metric is used that defines three levels of similarity (level0, level1, level2), similar to the two-player limit domain.
Given two betting sequences:

Table 14
The case representation used for producing case-based strategies in the domain of multi-player, limit Texas Hold'em.

Attribute	Type	Example
1. Hand Strength	Integer	1 – 50
2. Betting Sequence	String	*rcrrcc-crrcrrcc-ccr*, *rcc-ccc-ccrf*, ...
3. Board Texture	Class	*No-Salient*, *Flush-Possible*, *Straight-Possible*, *Flush-Highly-Possible*, ...
Action	Triple	(0.0, 0.95, 0.05), (0.6, 0.1, 0.3), ...
Outcome	Triple	(-∞, 9.0, -3.0), (-2.0, -4.5, -7.0), ...

level0: is satisfied if the two betting sequences exactly match one another.
level1: is satisfied if the number of bets/raises made by each active player is the same for each individual, non-current betting round.
level2: is satisfied if the total number of bets/raises made by each active player is the same for all non-current betting rounds combined.

For level1 and level2 similarity above, the current betting round must match exactly for two betting sequences to be considered similar. As in the two-player domain, similarity values of 1.0, 0.9 and 0.8 are assigned to the satisfied similarity level, respectively.

With Switching On the other hand, when *strategy switching* is enabled it is necessary to search a two-player case-base (when an opponent has folded). An inter-domain similarity metric is required that is able to determine similarity between the current three-player betting sequence and the two-player sequences contained within the case-base. We have already introduced the mapping used in this situation in Section 7.1.1. Here we provide an example that illustrates the result of this inter-domain mapping.
Consider the three-player betting sequence that takes place on the flop:

$$frc - r$$

The preflop action proceeds as follows: D folds, the SB raises and the BB calls. As a fold has occurred we can apply inter-domain mapping. The *player mapping* introduced in Fig. 9. tells us that:

$$SB \rightarrow ND$$
$$BB \rightarrow D$$

Hence, we require a two-player betting sequence that involves a single raise by the ND in the first round. The only legal two-player betting sequence that is appropriate, is as follows:

$$crc - r$$

Where, D checks, the ND raises and the D calls. As a sufficiently similar sequence has been found, future betting decisions would be based on the above two-player sequence.
Notice that the two-player betting sequence:

$$rc - r$$

which looks similar to the original three-player sequence ($frc - r$), would not be considered similar via the inter-domain mapping. While this sequence looks similar to the three-player sequence above, it nevertheless confuses the order in which bets/raises were made and would not accurately capture the important strength information that is captured by the mapping we have described. Finally, in cases where no sufficiently similar two-player betting sequence was located, similarity would be determined by reverting back to a search of the three-player case-base.

Table 15
Multi-player case-based strategy results with and without strategy switching. Values are in sb/h with 95% confidence intervals.

	Non-Switching	*Switching*
CaseBased3P	0.2101 ± 0.004	0.2218 ± 0.004
akuma	0.0898 ± 0.003	0.0875 ± 0.004
dpp	-0.2995 ± 0.004	-0.3093 ± 0.004

3. Board Texture: Similarity between board texture values is determined by the same similarity matrix used in previous domains, as specified in Fig. 3.

7.3. Experimental Results

We now present results obtained given the case-based strategies produced in the domain of three-player, limit Texas Hold'em. First, we evaluate the effect of *strategy switching* by comparatively evaluating *non-switching* strategies with their *switching* counterparts. Next, we present results from the 2011 Annual Computer Poker Competition, where our multi-player case-based strategies were challenged against some of the best multi-player limit agents in the world.

7.3.1. Strategy Switching Results

Here, we produced two sets of case-based strategies (one with *switching*, and one without) in order to determine the effect that strategy switching had on performance. Both strategies were challenged against the two multi-player, limit agents: *dpp* and *akuma*, which have been made publicly available by the Knowledge Engineering Group at Technische Universität Darmstadt [9]. The agent *dpp* is described as a "mathematically fair bot" that does no opponent modelling when making decisions, whereas *akuma* uses Monte-Carlo simulation with opponent modelling.

Both *non-switching* and *switching* case-based strategies were challenged against *dpp* and *akuma* for a total of 10 matches, where each match played consisted of 1000 duplicate hands. Recall from Section 3.2, that a three-player duplicate match involves six seat enumerations in order for each agent to experience the same game scenarios as their opponents, therefore each match involved playing 6000 hands. Finally, each of the 10 duplicate matches were seeded so that the scenarios encountered by the *non-switching* strategy were the same as those encountered by the *switching* strategy. This reduces the influence that the stochastic nature of the game has on the results. The final results are presented in small blinds won per hand with a 95% confidence interval in Table 15.

Firstly, notice that the case-based strategies produced (whether *switching* or *non-switching*) are able to beat both opponents at a significant level with 95% confidence. Second, the results show that multi-player case-based strategies *with switching* are able to achieve a greater overall profit than strategies that don't switch. The win rate difference observed is significant with 95% confidence. As long as the mapping does not destroy the information about the current game state, we are able to make use of the more detailed heads-up strategies, which positively affects performance. The results suggest that the inter-domain mapping we have defined is appropriate and can improve performance by making use of strategies that have previously been learned in a separate sub-domain.

7.3.2. 2011 AAAI Computer Poker Competition

We submitted a multi-player case-based strategy for the first time at the 2011 ACPC held at the Twenty-Fifth AAAI Conference on Artificial Intelligence. Our entry, Sartre3P, competed in the three-player limit Texas Hold'em competition and employed *strategy switching*, as described above. As *strategy switching* was used, Sartre3P was required to populate and search both three-player and two-player case-bases in order to make betting decisions. Both case-bases were populated by training on the hand history data of the winning agents from the previous year's total bankroll divisions (i.e. both two-player limit and three-player limit competitions). The results of the 2011 competition are given in Table 16. As with previous ACPC results, two winner determination procedures are displayed – *instant runoff* and *total bankroll*. Sartre3P placed 2^{nd} in the *instant runoff* division, out of 9 competitors. In this division Sartre3P was beaten only by the competitor Hyperborean.iro. For the *total bankroll* division, Sartre3P placed 1st and was the overall winner of this division. Sartre3P's average profit was

Table 16
2011 multi-player limit bankroll and runoff results. Values are in sb/h with 95% confidence intervals.

Multi-player Limit Bankroll Runoff	Multi-player Limit Total Bankroll	Average Won/Lost
1. Hyperborean-2011-3p-limit-iro	1. **Sartre3P**	0.266 ± 0.024
2. **Sartre3P**	2. Hyperborean-2011-3p-limit-tbr	0.171 ± 0.023
3. LittleRock	3. AAIMontybot	0.130 ± 0.045
4. dcubot3plr	3. LittleRock	0.122 ± 0.022
5. Bnold3	5. OwnBot	0.016 ± 0.035
6. AAIMontybot	6. Bnold3	-0.084 ± 0.028
7. OwnBot	7. Entropy	-0.099 ± 0.043
8. Entropy	8. player.zeta.3p	-0.521 ± 0.040
9. player.zeta.3p		

0.266 ± 0.024 sb/h, which was significantly greater than all other competitors in the competition.

7.3.3. Discussion

The results presented in Table 16 provide strong support for the efficacy of *case-based strategies* that have been trained on hand history data from previous year's competitions. Moreover, the results demonstrate that strategies produced in this *top-down* fashion actually have the capability to defeat their *bottom-up* counterparts. In particular, Sartre3P was trained on hand history data containing the decisions of the Hyperborean.tbr agent from the 2010 three-player, limit competition, yet the results from the 2011 competition show that the *top-down* strategy used by Sartre3P, together with *strategy switching*, was able to outperform the 2011 version of Hyperborean.tbr. While the differences between the 2010 and the 2011 versions of Hyperborean.tbr are not known, it is reasonable to expect that the quality of the 2011 version of Hyperborean.tbr is more likely to have improved (rather than degraded) over the year.

8. Conclusions

In conclusion, we have provided a comprehensive overview of our *case-based* strategies that employ a *top-down* approach by generalising decisions from a collection of data. We began with a description of the systematic maintenance performed on our strategies in the domain of two-player, limit Texas Hold'em. The final result was a framework we have employed to produce *case-based* strategies that have achieved top place finishes at international computer poker competitions, where the best computer poker agents in the world are challenged against each other. Using the lessons learned and insights obtained from the two-player, limit Hold'em domain, we extrapolated our framework to handle the more complicated domains of two-player, no-limit Texas Hold'em and multi-player, limit Texas Hold'em. In the no-limit domain, our case-based strategies produced the 2^{nd} best computer poker agent as judged by the results of the 2011 ACPC. In the multi-player, limit domain our agent was the winner of the total bankroll competition at the 2011 ACPC and achieved the greatest profit against all other competitors. These results show that the *case-based* frameworks we have presented are able to produce strong, sophisticated strategies that are competitive at an international level. Furthermore, we have shown that by augmenting *imitation*-based, *top-down* strategies with additional capabilities, such as *strategy switching*, it is possible to further improve the performance of the agents produced. This approach was used by our agent, which competed at the 2011 multi-player, limit Texas Hold'em competition. The results from this event show that, for the particular group of opponents challenged, our case-based agent was able to perform better than the expert whose decisions were used to train the strategy in the first place.

References

[1] ACPC. The Annual Computer Poker Competition, 2012. `http://www.computerpokercompetition.org/`.

[2] David W. Aha. Editorial. *Artificial Intelligence Review*, 11(1-5):7–10, 1997.

[3] David W. Aha, Matthew Molineaux, and Marc J. V. Ponsen. Learning to win: Case-based plan selection in a real-time strategy game. In *6th International Conference on Case-Based Reasoning, ICCBR 2005*, pages 5–20. Springer, 2005.

[4] Marv Andersen. Web posting at pokerai.org forums, general forums, August 2009. http://pokerai.org/pf3/viewtopic.php?t=2259&start=18.

[5] Rickard Andersson. Pseudo-optimal strategies in no-limit poker. Master's thesis, Umea University, 2006.

[6] Bryan Auslander, Stephen Lee-Urban, Chad Hogg, and Héctor Muñoz-Avila. Recognizing the enemy: Combining reinforcement learning with strategy selection using case-based reasoning. In *Advances in Case-Based Reasoning, 9th European Conference, ECCBR 2008*, pages 59–73, 2008.

[7] Darse Billings, Neil Burch, Aaron Davidson, Robert C. Holte, Jonathan Schaeffer, Terence Schauenberg, and Duane Szafron. Approximating game-theoretic optimal strategies for full-scale poker. In Georg Gottlob and Toby Walsh, eds., *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 661–668. Morgan Kaufmann, 2003.

[8] Darse Billings, Aaron Davidson, Terence Schauenberg, Neil Burch, Michael H. Bowling, Robert C. Holte, Jonathan Schaeffer, and Duane Szafron. Game-tree search with adaptation in stochastic imperfect-information games. In H. Jaap van den Herik, Yngvi Björnsson, and Nathan S. Netanyahu, eds., *Computers and Games, 4th International Conference, CG 2004*, pages 21–34. Springer, 2004.

[9] Technische Universität Darmstadt. Computer poker bots, 2012. http://www.ke.tu-darmstadt.de/resources/poker.

[10] Ian Fellows. Artificial Intelligence poker, 2012. http://www.deducer.org/pmwiki/index.php/.

[11] Michael W. Floyd, Babak Esfandiari, and Kevin Lam. A case-based reasoning approach to imitating robocup players. In *Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society Conference*, pages 251–256, 2008.

[12] PokerAI.org Forums. Fell Omen 2, 2011. http://pokerai.org/pf3/viewtopic.php?f=3&t=3593.

[13] Andrew Gilpin, Samid Hoda, Javier Peña, and Tuomas Sandholm. Gradient-based algorithms for finding Nash equilibria in extensive form games. In Xiaotie Deng and Fan Chung Graham, eds., *Internet and Network Economics, Third International Workshop*, pages 57–69. Springer, 2007.

[14] Andrew Gilpin and Tuomas Sandholm. Better automated abstraction techniques for imperfect information games, with application to Texas Hold'em poker. In Edmund H. Durfee, Makoto Yokoo, Michael N. Huhns, and Onn Shehory, eds., *6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007)*, pages 192–200. IFAAMAS, 2007.

[15] Andrew Gilpin, Tuomas Sandholm, and Troels Bjerre Sørensen. A heads-up no-limit Texas Hold'em poker player: discretized betting models and automatically generated equilibrium-finding programs. In Lin Padgham, David C. Parkes, Jörg P. Müller, and Simon Parsons, eds., *7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 911–918. IFAAMAS, 2008.

[16] Kristian J. Hammond. Case-based planning: A framework for planning from experience. *Cognitive Science*, 14(3):385–443, 1990.

[17] Michael Johanson, Martin Zinkevich, and Michael H. Bowling. Computing robust counter-strategies. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, eds., *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada,*. Curran Associates, Inc., 2007.

[18] Michael Bradley Johanson. Robust strategies and counter-strategies: Building a champion level computer poker player. Master's thesis, University of Alberta, 2007.

[19] Janet Kolodner. *Case-Based Reasoning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[20] Kinshuk Mishra, Santiago Ontañón, and Ashwin Ram. Situation assessment for plan retrieval in real-time strategy games. In Klaus-Dieter Althoff, Ralph Bergmann, Mirjam Minor, and Alexandre Hanft, eds., *Advances in Case-Based Reasoning, 9th European Conference, ECCBR 2008*, volume 5239 of *Lecture Notes in Computer Science*, pages 355–369. Springer, 2008.

[21] Yu. Nesterov. Excessive gap technique in nonsmooth convex minimization. *SIAM J. on Optimization*, 16(1):235–249, 2005.

[22] Santiago Ontañón, Kinshuk Mishra, Neha Sugandh, and Ashwin Ram. Case-based planning and execution for real-time strategy games. In Rosina Weber and Michael M. Richter, eds., *7th International Conference on Case-Based Reasoning, ICCBR 2007*, volume 4626 of *Lecture Notes in Computer Science*, pages 164–178. Springer, 2007.

[23] Santiago Ontañón, Kinshuk Mishra, Neha Sugandh, and Ashwin Ram. On-line case-based planning. *Computational Intelligence*, 26(1):84–119, 2010.

[24] Jay H. Powell, Brandon M. Hauff, and John D. Hastings. Utilizing case-based reasoning and automatic case elicitation to develop a self-taught knowledgeable agent. In *Proceedings of the Workshop on Challenges in Game AI, Nineteenth National Conference on Artificial Intelligence (AAAI-2004), San Jose, California, USA*. AAAI Press, 2004.

[25] PokerAI.org Research. Pokerftp hand history database, March 2011. http://pokerftp.com/.

[26] Christopher K. Riesbeck and Roger C. Schank. *Inside Case-Based Reasoning*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1989.

[27] Nicholas Abou Risk and Duane Szafron. Using counterfactual regret minimization to create competitive multiplayer poker agents. In Wiebe van der Hoek, Gal A. Kaminka, Yves Lespérance, Michael Luck, and Sandip Sen, eds., *9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 159–166. IFAAMAS, 2010.

[28] Jonathan Rubin and Ian Watson. Similarity-based retrieval and solution re-use policies in the game of texas hold'em. In *18th International Conference on Case-Based Reasoning.*, pages 465–479. Springer-Verlag, 2010.

[29] Jonathan Rubin and Ian Watson. On combining decisions from multiple expert imitators for performance. In Toby Walsh, ed., *IJCAI-11, Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 344–349. IJCAI/AAAI, 2011.

[30] David Schnizlein, Michael H. Bowling, and Duane Szafron. Probabilistic state translation in extensive games with large action sets. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 278–284, 2009.

[31] Manu Sharma, Michael P. Holmes, Juan Carlos Santamaría, Arya Irani, Charles Lee Isbell Jr., and Ashwin Ram. Transfer learning in real-time strategy games using hybrid CBR/RL. In Manuela M. Veloso, ed., *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1041–1046. IJCAI/AAAI, 2007.

[32] David Sklansky. *The Theory of Poker*. Two Plus Two Publishing, Las Vegas, Nevada, 2005.

[33] David Sklansky and Mason Malmuth. *Hold'em Poker For Advanced Players*. Two Plus Two Publishing, Las Vegas, Nevada, 1994.

[34] Neha Sugandh, Santiago Ontañón, and Ashwin Ram. On-line case-based plan adaptation for real-time strategy games. In Dieter Fox and Carla P. Gomes, eds., *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008*, pages 702–707. AAAI Press, 2008.

[35] K. Sycara, D. Navin Chandra, R. Guttal, J. Koning, and S. Narasimhan. Cadet: A case-based synthesis tool for engineering design. *International Journal of Expert Systems*, 4(2):157–188, 1991.

[36] Guy Van den Broeck, Kurt Driessens, and Jan Ramon. Monte-carlo tree search in poker using expected reward distributions. In Zhi-Hua Zhou and Takashi Washio, eds., *Advances in Machine Learning, First Asian Conference on Machine Learning, ACML 2009*, volume 5828 of *Lecture Notes in Computer Science*, pages 367–381. Springer, 2009.

[37] Ian Watson. *Applying Case-Based Reasoning : Techniques for Enterprise Systems*. San Francisco, Calif. : Morgan Kaufmann, 1997.

[38] Kevin Waugh, David Schnizlein, Michael H. Bowling, and Duane Szafron. Abstraction pathologies in extensive games. In Carles Sierra, Cristiano Castelfranchi, Keith S. Decker, and Jaime Simão Sichman, eds., *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, pages 781–788. IFAAMAS, 2009.

[39] Martin Zinkevich, Michael Johanson, Michael H. Bowling, and Carmelo Piccione. 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*, 2007.