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

*Jonathan Rubin & Ian Watson*
*University of Auckland*

http://www.cs.auckland.ac.nz/research/gameai

# Introduction

## SARTRE

- Presented SARTRE system overview last year at CBR for Computer Games Workshop
- Uses CBR to play Texas Hold'em
- Competed in 2009 IJCAI Computer Poker Competition
- Since then Sartre has undergone case-based maintenance

# Introduction

## 1. Outcomes of maintenance

Present major outcomes of maintenance phase

## 2. Solution re-use policies

Introduce and evaluate 3 policies for re-using decisions

## 3. Experimental results

Self-play experiments

*Pre* & *post* maintenance systems

# Introduction

## Claim 1

Modifying the solution representation results in changes to the problem coverage

## Claim 2

Different policies for re-using solutions leads to changes in performance

# Overview

- Background

  Types of strategies

  Related approaches

  CBR Motivation
- Our Approach

  Highlighting major differences between pre & post maintenance systems.
- Experimental results

  Provide justification for claims 1 & 2
- Conclusions

# Poker Strategies

# A Poker Strategy

- At every decision point a probability triple is required that indicates the proportion of the time a player should either fold, call or raise

$$(f,c,r) \rightarrow (0,\ 0.5,\ 0.5)$$

# Types of Strategies

- Nash Equilibria
  - Robust strategies that attempt not to lose to any type of opponent
- Exploitive Strategies
  - Attempts to react to an opponent's play in a way that allows maximum exploitability of that opponent
  - Requires opponent modeling

# Rock-Paper-Scissors Example

- Nash equilibrium
  - *(R,P,S) → (1/3, 1/3, 1/3)*
  - The Nash player will never lose against any player in the long run
- Along comes Jimmy who only ever plays Paper

# Rock-Paper-Scissors Example

- The Nash player will continue to play
  - *(1/3, 1/3, 1/3)*
  - Lose 33%, Win 33%, Draw 33%
  - The Nash player will still only draw against Jimmy

# Rock-Paper-Scissors Example

- However because we know Jimmy's strategy, an exploitive player would be better off using the strategy
  - *(0, 0, 1)*
  - i.e. a best response that maximally exploits Jimmy at every decision point
- Now, against Jimmy the exploitive player will win
  - Consequence is that the exploitive player plays off the equilibrium, and is hence subject to potential exploitation itself

# *e*-Nash Equilibrium

- A Nash equilibrium can easily be computed for Rock-Paper-Scissors
- However, the poker game tree is much to large to find exact Nash equilibria
  - Abstractions required
- Can only approximate Nash-equilibria
  - *e*-Nash Equilibria
  - *e* specifies a lower bound on how exploitable the equilibrium strategy is

# Approaches

- Game theoretic approaches
  - Linear Programming
  - Repeated Play
- Dynamic Tree Search
- Rule-Based Systems
- Evolutionary algorithms
- Artificial Neural Networks
- Bayesian networks
- Case-based reasoning

# CBR Motivation

- Expert imitation

  Approximate the play of an 'expert' or group of 'experts' via observation and generalisation.

  Different case-bases can model different types of players, avoiding the creation of complicated mathematical models

- *e*-Nash Equilibrium approximation

  Determine how closely a large *e*-Nash equilibrium strategy can be approximated with a compact case-base that relies on finding similar cases and generalising the observed actions

# Approach Overview

# Approach Overview

## Overview

- Cases are attribute-value pairs
- Separate case-bases are used for each stage of game
- When a decision is required a case is created to describe the current state of the game and the appropriate case-base is searched to find similar cases
- The solution of the similar cases are reused for the current situation

# Case Representation

- Features

| Attribute | Type | Example |
|---|---|---|
| **1. Hand Type** | Class | *Missed, Pair, Two-Pair, Set, Flush,Flush-Draw, Straight-Draw, …* |
| **2. Betting Sequence** | String | *rc-c, crrc-crrc-cc-r, …* |
| **3. Board Texture** | Class | *No-Salient, Flush-Possible, Straight-Possible, Flush-Highly-Possible, …* |

# Maintenance Outcomes

# Maintenance Outcomes

- Knowledge containers updated
  - Case-base knowledge
  - Retrieval knowledge
- *Pre*-maintenance
  - Sartre-1
- *Post*-maintenance
  - Sartre-2

# Similarity-Based Retrieval

- *k*NN
- Sartre-1

  Required absolute matches

  *k* could vary between 0 – *N*, where *N* is the number of cases in the case-base.

  If *k* = 0, then default policy = always-call
- Sartre-2

  Updated retrieval knowledge

  *k* = 1

  No default policy

# Solution Representation

- Maintenance phase resulted in updates to solution representation
- Sartre-1

  Trained by recording decisions of 'expert' player

  For each observed decision there exists 1 case

# Solution Representation

- Sartre-1

<u>Case 1</u>

**Features:**

1. Hand strength: *flush*
2. Betting sequence: *c*
3. Board Texture: *fp*

**Solution:**

Action: r
Outcome: 6

# Solution Representation

- Sartre-1

**Case 1**

**Features:**

1. Hand strength: *flush*
2. Betting sequence: *c*
3. Board Texture: *fp*

**Solution:**

Action: r
Outcome: 6

**Case 2**

**Features:**

1. Hand strength: *flush*
2. Betting sequence: *c*
3. Board Texture: *fp*

**Solution:**

Action: r
Outcome: -4

# Solution Representation

- Sartre-1

**Case 1**

**Features:**

1. Hand strength: *flush*
2. Betting sequence: *c*
3. Board Texture: *fp*

**Solution:**

Action: r
Outcome: 6

**Case 2**

**Features:**

1. Hand strength: *flush*
2. Betting sequence: *c*
3. Board Texture: *fp*

**Solution:**

Action: r
Outcome: -4

**Case 3**

**Features:**

1. Hand strength: *flush*
2. Betting sequence: *c*
3. Board Texture: *fp*

**Solution:**

Action: c
Outcome: 2

# Solution Representation

- Sartre-1

**Case 1**

**Features:**

1. Hand strength: *flush*
2. Betting sequence: *c*
3. Board Texture: *fp*

**Solution:**

Action: r
Outcome: 6

**Case 2**

**Features:**

1. Hand strength: *flush*
2. Betting sequence: *c*
3. Board Texture: *fp*

**Solution:**

Action: r
Outcome: -4

**Case 3**

**Features:**

1. Hand strength: *flush*
2. Betting sequence: *c*
3. Board Texture: *fp*

**Solution:**

Action: c
Outcome: 2

Action: (0.0, 0.66, 0.33)

Outcome: (-*inf*, 1.0, 2.0)

# Solution Representation

- Sartre-2

  Updated solution representation

    Action Triple / Outcome Triple

  Pre-process the training data

  No need to retain duplicate cases

  Compact case-base

# Solution Representation

- Sartre-2

**Case 1**

**Features:**

1. Hand strength: *flush*
2. Betting sequence: *c*
3. Board Texture: *fp*

**Solution:**

Action: (0.0, 0.66, 0.33)
Outcome: (-*inf*, 1.0, 2.0)

# Solution Representation

- Case-base size

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

# Solution Representation

- Claim 1

  *“Changes in solution representation results in changes to problem coverage”*

- Justification (via argument)

  Training period for Sartre-1 had to be cut short due to case storage & memory requirements

  Sartre-2's updated solution representation allows a much more compact case-base

  These reduced costs allow a larger set of training data to be used to train Sartre-2

  Sartre-2 encounters and stores solutions for a wider range of problems than Sartre-1.

# Solution Re-use Policies

# Solution Re-use Policies

- Once similar case(s) retrieved a betting decision is required
- 3 Policies

    *Probabilistic*

    *Majority-rules*

    *Best-outcome*

# Solution Re-use Policies

- 3 Policies

    *Probabilistic*: probabilistically mix between the actions

*Action*: ( 0.0, 0.66, 0.33 )

# Solution Re-use Policies

- 3 Policies

  *Majority rules*: re-use the action taken the majority of the time

*Action*: ( 0.0, 0.66, 0.33 )

# Solution Re-use Policies

- 3 Policies

  *Majority rules*: re-use the action taken the majority of the time

  *Action*: ( 0.0, $\boxed{0.66,}$ 0.33 )

# Solution Re-use Policies

- 3 Policies

  *Best outcome*: re-use the decision with the best recorded outcome

*Outcome*: ( *-inf*, 1.0, 2.0 )

# Solution Re-use Policies

- 3 Policies

*Best outcome*: re-use the decision with the best recorded outcome

*Outcome*: ( *-inf*, 1.0, $\boxed{2.0}$ )

# Experimental Results

# Experiments

- Duplicate Matches
  - *N* hands in forward + backwards direction
  - Set of hands played
  - Set of hands replayed, but agents receive the cards that their opponent previously received
  - Reduces variance
- Small bets per hand (sb/h)

# Experiment 1

- Solution re-use policies
- Claim 2

  *“Different policies for re-using solutions leads to changes in performance”*

# Experiment 1

- Solution re-use policies
- Method

    Each policy plays 3 duplicate matches against each other

    $N$ = 3000

    18,000 hands in total played between each policy

# Experiment 1

- Solution re-use policies
- Results

All matches statistically significant

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

# Experiment 1

- Solution re-use policies
- Discussion

  Support for claim 2

  Similar outcome observed against other opponents

  Best-outcome: good outcomes don't necessarily represent good betting decisions

# Experiment 1

- Solution re-use policies
- Discussion

  Majority rules performs best.

  Probably due to type of opponent (Nash-based)

  Nash-based opponents only win when the opponent makes a mistake

  Majority-rules biases actions to non-exploratory actions (which are less likely to be dominated errors)

  Nash-based opponents won't exploit biased action selection

# Experiment 2

- *Pre & Post* Maintenance
- Method

  Sartre-1 Vs. Sartre-2

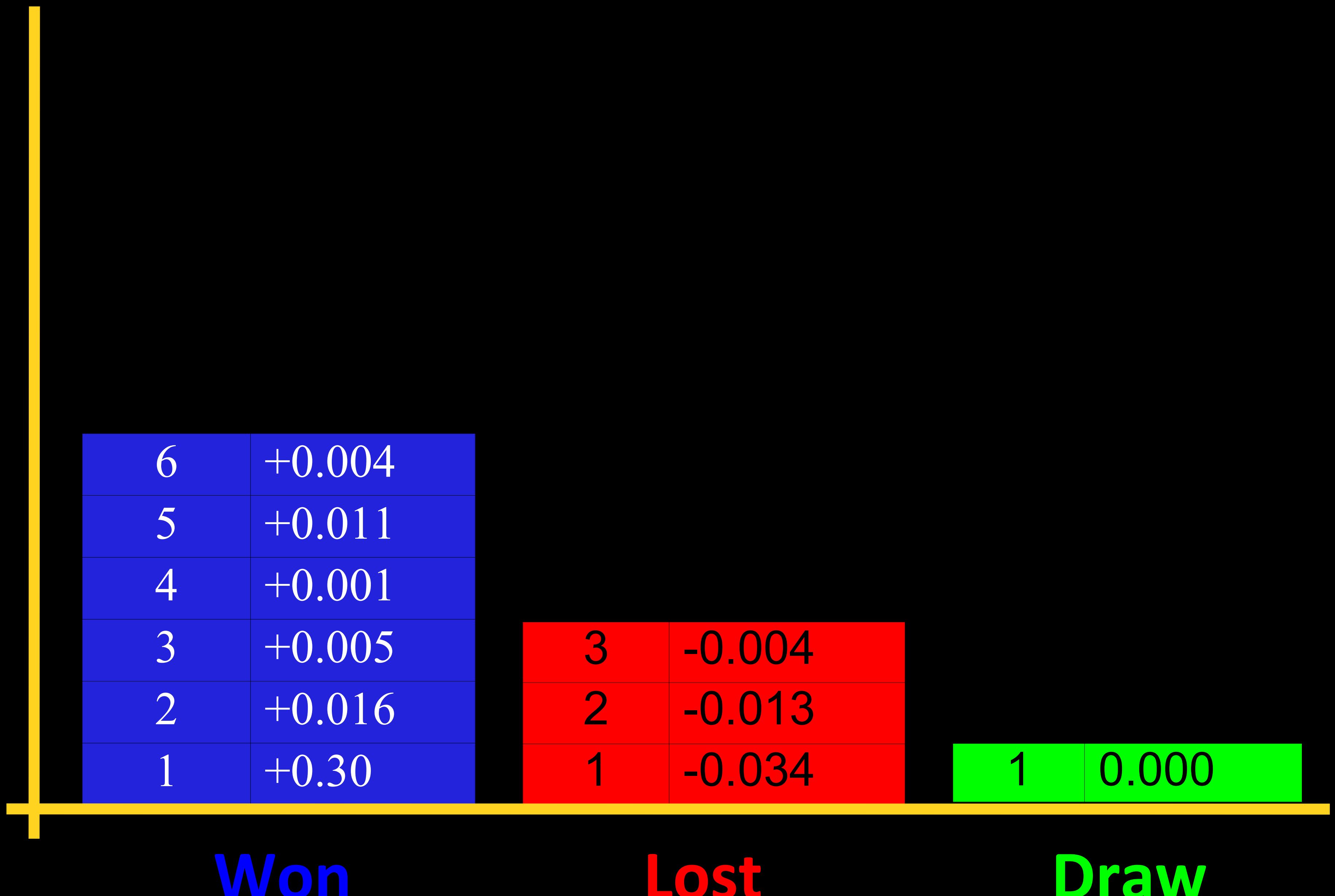
  Both use majority-rules policy

  10 duplicate matches

  $N$ = 3000

  60,000 hands played total

# Sartre-2 Vs. Sartre-1 (sb/h)



# Experiment 2

- *Pre & Post* Maintenance
- Results

**Sartre-2**

| | |
|---|---|
| Mean: | 0.0286 |
| Std dev: | 0.096368 |
| 95% CI: | [-0.04034, 0.09754] |

# Experiment 2

- *Pre & Post* Maintenance
- Discussion
    - Sartre-2 performs a little better than Sartre-1
    - Results against other agents support this finding
    - Many factors that could contribute to difference
        - Improved retrieval knowledge
        - Updated case-base knowledge
        - Improved problem coverage etc...

# Annual Computer Poker Competition Results Summary

| Agent | Division | Rank |
|---|---|---|
| Sartre-1 (2009) | Bankroll | **6th out of 12** |
| | Equilibrium | **7th out of 13** |
| Sartre-2 (2010) | Bankroll | **3rd out of 13** |
| | Equilibrium | **6th out of 13** |

# Conclusions

- Outcomes of maintenance phase

  Justified claim 1 (via argument)

- Solution re-use policies

  Justified claim 2 (via comparative evaluation)

Thank you!

To challenge Sartre go to:

*www.cs.auckland.ac.nz/poker*