Real-Time Opponent Modelling in Trick-Taking Card Games

Jeffrey Long and **Michael Buro**
Department of Computing Science, University of Alberta
Edmonton, Alberta, Canada T6G 2E8
{*jlong*1 | *mburo*}@cs.ualberta.ca

Abstract

As adversarial environments become more complex, it is increasingly crucial for agents to exploit the mistakes of weaker opponents, particularly in the context of winning tournaments and competitions. In this work, we present a simple post processing technique, which we call Perfect Information Post-Mortem Analysis (PIPMA), that can quickly assess the playing strength of an opponent in certain classes of game environments. We apply this technique to skat, a popular German card game, and show that we can achieve substantial performance gains against not only players weaker than our program, but against stronger players as well. Most importantly, PIPMA can model the opponent after only a handful of games. To our knowledge, this makes our work the first successful example of an opponent modelling technique that can adapt its play to a particular opponent in real time in a complex game setting.

1 Introduction

Opponent modelling is the problem of predicting the actions of other agents in an adversarial environment. Classically, the general implicit approach to opponent modelling has been to assume that all players are attempting to play a *Nash equilibrium* — a game-theoretically optimal policy that guards against the worst case. Such an assumption is critical to the alpha-beta minimax search techniques that have been applied with great success to perfect information games like chess and checkers.

However, when adversarial environments become more complex, notably through the inclusion of stochasticity and imperfect information, it becomes more important to consider different opponent models for two reasons. The first reason is that optimal strategies in these environments are often overly defensive — they guarantee that we cannot lose much, but also that we cannot gain much either, no matter the actions of the opponent. In settings such as competitive tournaments, it is often not enough to play well against strong opposition; potential tournament winners must also maximize their gains against weaker, more exploitable competitors. The second reason is that in imperfect information environments, it is often helpful to infer hidden state variables based on actions of other agents. This, of course, requires an accurate model of how agents behave in the environment. Thus, an appropriate opponent model allows for an agent to explicitly exploit opponent weaknesses, and to improve its own decision making through superior inference of hidden information.

In this paper, we describe a simple post-processing analysis technique that can be used to assess an agent's general decision quality in certain classes of environments. We then apply this technique to a computer player for the game of skat, Germany's national card game, and show how the agent is able to appropriately adapt its play against varying strengths of opponents. Most importantly, this adaptation is fast enough to be performed in real time and requires only a handful of encounters with a particular opponent in order to be effective.

2 Background and Related Work

Opponent modelling in adversarial settings has received some recent attention particularly in imperfect information games, where inference of hidden information and opponent exploitation are both problems of interest.

On the inference front, in the game of Scrabble, [Richards and Amir, 2007] describe a method of inferring letters on the opponent's rack in order to bias letter distributions used in simulations. Equipped with this inference technique, Quackle, which at the time was the strongest Scrabble program and had previously beaten a former human World-champion, achieved a substantial 5.2 point score advantage over the original Quackle version. Similarly, in the card game of skat, [Buro *et al.*, 2009] report that card-play search techniques benefit considerably from shaping hand distributions based on opponents' bidding and game declarations.

Opponent modelling has also been of keen interest to the rapidly growing body of recent computer poker research. When research in this field first began in earnest in 2001, the critical need for opponent modelling was cited as a key feature that distinguished poker from more traditional perfect information games [Billings *et al.*, 2002]. However, although early forays into poker opponent modelling do exist [Davidson *et al.*, 2000] [Southey *et al.*, 2005], so far the most successful approach to computer poker has been to pre-compute game-theoretically optimal strategies in smaller, abstract versions of the game and subsequently map these

strategies back into the full game [Zinkevich *et al.*, 2008] [Gilpin and Sandholm, 2006]. Indeed, an examination of the top contenders of the 2010 AAAI Computer Poker competition (`http://www.computerpokercompetition.org/`) shows that the clear majority are static players with no online adaptation to a particular opponent.

There has, however, been some recent exciting progress on this front. In 2009, [Johanson and Bowling, 2009] proposed Data Biased Response (DBR), an opponent modelling technique that balances an agent's ability to exploit an opponent while avoiding being exploited itself. Most impressively, DBR comes with theoretical guarantees that the trade-off it achieves between opponent-exploitation and self-exploitability is optimal. However, there are two significant caveats: first, DBR still requires on the order of thousands of observations to begin significantly exploiting the opponent, and second, computing the DBR strategy itself once given its observations is much too slow to be done in real time. It is fair to say that real-time, online opponent modelling remains an open question in computer poker research.

3 Methodology

In his seminal work on building a champion-calibre computer bridge player, Ginsberg remarks that both human bridge experts and his program GIB have an extraordinarily low "double-dummy mistake rate" [Ginsberg, 2001]. What he means by this is that if one were to compute the *perfect information value* of the game — the final resulting game value under the assumption that the location of all cards is known to all players, and all players play perfectly from that point onwards — that value would rarely change following any play by a human expert or by his program.

Ginsberg does not dwell on this observation, other than to consider it as supporting evidence that his program GIB has reached or surpassed human expert level playing strength. For us, this observation will form the foundation of our modelling approach, which we describe below.

3.1 Perfect Information Post-Mortem Analysis

We will refer to our approach as Perfect Information Post-Mortem Analysis, or PIPMA. The procedure is intended to be called at the conclusion of each episode in an environment of interest and to incrementally update a *mistake rate* for each agent involved in the episode. This is done by examining each state in which an agent acted and computing the perfect information value of that state before and after the agent's move. If there is a difference, the agent's move is flagged as a mistake. It is also necessary to determine whether it was even possible to make a mistake, which in the worst case, requires examining every available move. States where a mistake was not possible are discarded from the analysis. The agent's final mistake rate is defined as the ratio of mistakes to mistake opportunities. Pseudo-code for the PIPMA process is shown in Figure 1.

We note that PIPMA requires two assumptions about the environments to which it can be applied. The first is that by the end of each environmental episode, all previously hidden information from the environment must have become public for all agents. The second is that perfect information values must be relatively easy to compute for all states in the environment. These assumptions hold in most trick-taking card games, such as bridge as studied by Ginsberg, and skat, the test domain for our work here.

Algorithm 1 PIPMA post-game update procedure for a single player. Winning positions have value 1 and losing positions, value -1; we assume here that there are no other values.

Require: completed episode E, agent p
  for each state s in E for which p is to move **do**
    $Win \leftarrow \emptyset$
    $Lose \leftarrow \emptyset$
    for each action a available in s **do**
      compute value(a)
      {Returns the Perfect Information game value of selecting a at s}
      if $value(a) = 1$ **then**
        $Win \leftarrow Win \cup \{a\}$
      else
        $Lose \leftarrow Lose \cup \{a\}$
      end if
    end for
    if $Win \neq \emptyset$ AND $Lose \neq \emptyset$ **then**
      $mistakeOpps(p) \leftarrow mistakeOpps(p) + 1$
      {p's opportunies to make mistakes, maintained globally}
      if p's chosen action in $s \in Lose$ **then**
        $mistakes(p) \leftarrow mistakes(p) + 1$
        {p's total mistakes made, maintained globally}
      end if
    end if
  end for

3.2 Theoretical and Empirical Properties of PIPMA

PIPMA assumes that the actions it identifies as mistakes in the perfect information version of an environment are actually undesirable in the original imperfect information setting and should be avoided by rational players. It is simple to show that there is no theoretical guarantee that this will be the case. In fact, PIPMA can, in principle, consider an optimal move to be a perfect information mistake. We demonstrate this through means of the example shown in Figure 1.

In this example, there are two players, MAX (represented by circles) and MIN (represented by squares), who are playing an imperfect information game. There are four possible worlds, determined secretly at the start of the game. MAX is to move at the root and can go either right or left. If she goes left, then she is to move again. Now she has four options, each one of which wins in exactly one distinct world and loses in the other three. Because MAX does not know which is the true world, she must simply pick a move at random, and achieve a payoff of 1 with probability 0.25 and -1 otherwise. Alternatively, if MAX goes right at the root, she can foist the guess onto MIN, and therefore achieve the positive payoff with probability 0.75. Clearly, then, MAX's best move is to go to the right. However, from a perfect informa-

tion perspective, where all players know the true world, going to the right is a mistake for MAX, since MIN has no need to "guess" in this game's perfect information variant. Therefore, MAX moving right will be flagged as a mistake by PIPMA, even though it is the clear optimal move in the real game.

In spite of this theoretical caveat, it seems that pathological cases like those shown in Figure 1 rarely occur, at least in the environments where PIPMA has been examined. Ginsberg shows that in bridge, human world experts have a PIPMA mistake rate of approximately 2% while leading the first trick of the game, and this rate goes steadily down as the game progresses. In skat, we measured card-play mistake rates for a collection of sixty-five randomly selected human players, each of whom had played at least 2000 games on an online skat server. We plot these mistake rates against the players' average *score-per-list* — the standard scoring metric in skat — in Figure 2. Although, as we will see below, card-play strength is only one aspect of playing good skat, the general trend is clear: higher-scoring players tend to have lower PIPMA mistake rates. We also show skat data in Figure 3 indicating how PIPMA mistake error declines as a function of the number of tricks played so far. On this plot, Kermit is a computer player by [Buro *et al.*, 2009] that we will discuss further below, Ron Link is a high-ranked human expert, and the third player is an arbitrarily selected amateur human player. Although overall mistake rates appear to be higher in skat than in bridge, we see a trend similar to Ginsberg's observation, namely that the mistake rate declines as the game progresses.

Finally, we note that PIPMA is relatively difficult for a canny human (or other opponent-modelling computer) player to exploit, so long as cases such as described in Figure 1 are indeed rare (and it is likely inadvisable to use PIPMA at all in environments where they are not). In order to be unduly "underestimated" by PIPMA, a player must intentionally throw away games that, from a perfect information perspective, she has already won. It seems exceedingly unlikely that forfeiting guaranteed wins frequently enough to cause PIPMA to make a serious misestimation of a player's skill could result in a significant long-term advantage. However, this is true if PIPMA flags only mistakes that change the game value from a win to loss. If, for example, PIPMA considers a move that loses points according to some heuristic function — say, failing to maximize tricks taken in bridge beyond what is needed to win the contract — to be a "mistake," then it could become vulnerable to an exploitative opponent. On the other hand, if most of these mistakes are genuine, flagging them with PIPMA could decrease the amount of data to form an accurate opinion of the opponent's mistake rate.

3.3 Applying Perfect Information Post-Mortem Analysis to Skat

In this section we describe how we use the PIPMA technique described above to add a real-time opponent modelling component to a computer player for the game of skat.

Skat is a trick-taking card game for three players. It is most widely played in Germany, but skat clubs exist all over the world. For the purposes of this paper, the full rules of the game are not important; what is important is that a hand of

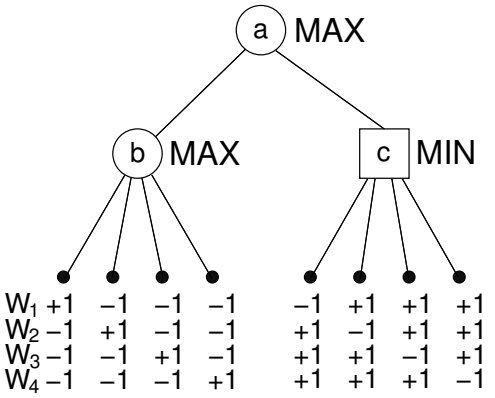

Figure 1: A synthetic zero-sum game in which PIPMA flags the optimal move (a-c) as a mistake. The payoffs at the leaves in each world W_i are given in view of player MAX.

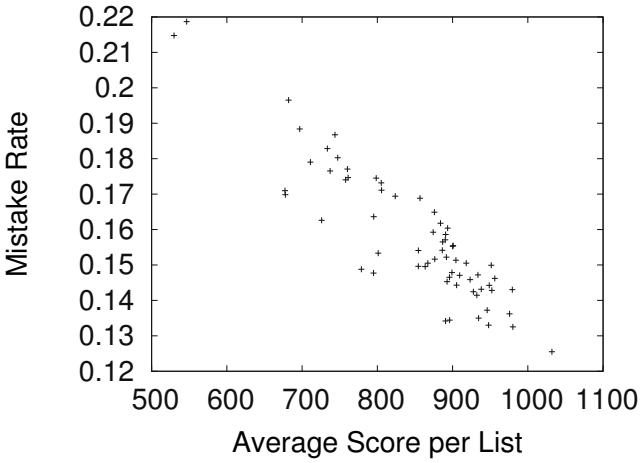

Figure 2: PIPMA mistake rate plotted against average score-per-list for sixty-five human skat players.

skat consists of two phases, bidding and then card-play. During the bidding phase, players compete in an auction to become the *soloist* for the duration of the hand. The soloist has the potential to gain many points but can also lose points as well, so it is an inherently risky position. The two players who lose the bid become *defenders* and form a temporary coalition against the soloist for the card-play phase. The defenders can win a small number of points if they defeat the soloist during card-play, but cannot lose points in any way. The objective of all players is to score as many points as possible over a series of hands. For more complete information on the game rules, we refer the interested reader to `http://www.davidparlett.co.uk/skat`.

As a basis for our computer skat player, we will use a player called Kermit, as described by [Buro *et al.*, 2009]. Kermit uses two distinct techniques to deal with the two different phases of skat. For card-play, Kermit uses Perfect Information Monte Carlo (PIMC) search, as first used by Ginsberg in bridge [Ginsberg, 2001]. For bidding, Kermit uses

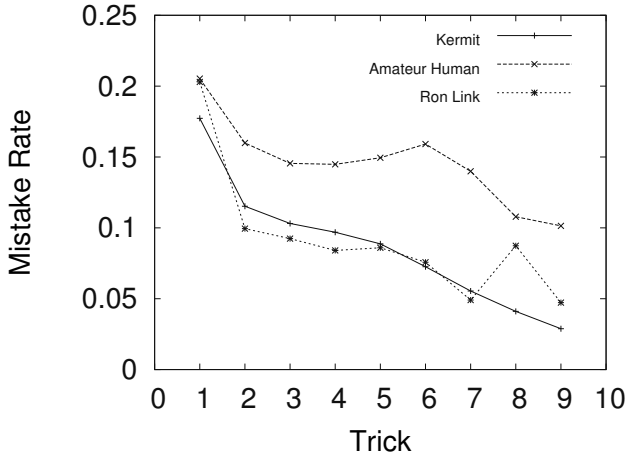

Figure 3: PIPMA mistake rate plotted as a function of the current trick in skat for three players. Kermit is a computer player, Ron Link is a high-ranked human expert and the third player is an arbitrary human of roughly average strength.

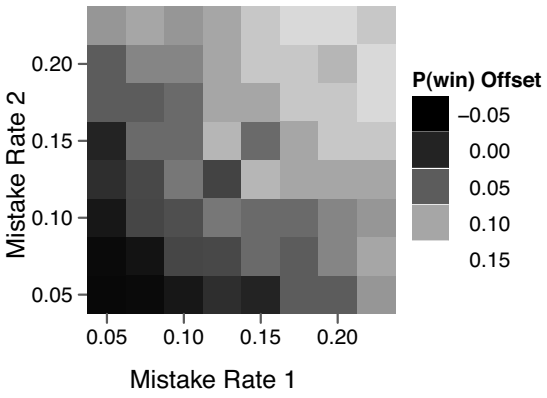

Figure 4: Synthetic tournament results displaying the average P(win) estimation error in Kermit's bidding-phase hand evaluator, plotted as a function of the mistake rates of Kermit's two opponents.

a static evaluation function learnt from millions of games of human data that assigns to a hand a winning probability for the soloist. If this probability is greater than a static threshold, then Kermit continues to bid, otherwise, the program passes.

Our new program, which we will refer to as the Count (perhaps due to the program's manic obsession with counting points and moves of the other players), is identical to Kermit, except that after every hand, it post-processes the card-play moves of all players involved using PIPMA and updates their mistake rates. This post-processing takes approximately one second after each game, and is therefore practical even for real-time play against humans. The Count then uses this information to dynamically adjust its bidding threshold during the bidding phase of future games.

The basic intuition behind this threshold adjustment is simple: bid more aggressively against weak players (players with a high mistake rate), and less aggressively against stronger ones. The problem that remains is exactly how much to adjust this bidding threshold based on the observed mistake rates of the opponents. To gain insight into this decision, we constructed a synthetic skat player that is directly defined in terms of its mistake rate m as follows: Whenever the player is to move, it is explicitly informed of the true world, and computes the sets of winning and losing moves for the current (perfect information) position. Then, assuming both sets are non-empty, with probability $1 - D(m)$ the player chooses a winning move to play, using Kermit's PIMC search procedure (and thus for this purpose ignoring that the player knows the true world) to break ties if necessary. $D(m)$ is a function that maps the player's global mistake rate to a local one that is dependent on the current stage of the game. With probability $D(m)$, the player is *forbidden* from selecting a winning move and must play a losing move (e.g. make a mistake) instead, again using PIMC search to break ties among candidate losing moves. We chose $D(m) = m + 0.07$ for the game's first trick and $D(m) = \max(0, (m - (0.01(trick - 1))))$ for subsequent tricks. These choices were based on the approximate observed rate at which Kermit's own mistake rate declined as a function of the current trick, and so that empirically the synthetic player's global mistake rate ends up being very close to m.

We then played a series of games between Kermit and synthetic players of varying strength. For each series, we record the average error in Kermit's bidding evaluation function over all games in which Kermit played as soloist, where the error for an individual game G is defined as follows:

$$\text{error}(G) = \text{win}(G) - \text{pWin}(G) \quad (1)$$

$\text{win}(G)$ is 1 if Kermit won in G as soloist and 0 otherwise, while $\text{pWin}(G)$ is the winning probability for G as computed by Kermit's evaluation function. The results of these experiments are shown in Figure 4.

Note that, while useful, these synthetic results do not take account of more complex factors, such as a player's potential to win extra points as a defender due an opponent's high error rate. However, using these as a starting point, we performed some empirical tuning and ultimately chose the formula shown in Equation (2) to use for the Count, where $m(p_1, p_2)$ is simply the average of the individual mistake rates of the two opponents, p_1 and p_2. Recall that this is an offset to the Count's bidding threshold, so that a positive offset increases the threshold, making the program more conservative, while a negative offset decreases the threshold, making the program bid on riskier games.

$$\text{offset}(p_1, p_2) = \begin{cases} 0.1 - m(p_1, p_2) & \text{if } m(p_1, p_2) \leq 0.1 \\ 0.05 - 0.5m(p_1, p_2) & \text{otherwise} \end{cases} \quad (2)$$

4 Experimental Results

4.1 Computer Player Tournaments

To assess the strength of our PIPMA modelling approach in a real game environment, we ran a series of skat tournaments

Table 1: Tournament results between players, all performed on the same set of 2000 deals. In each match, the column player played against two clones of the row player. We report the average score-per-list difference (row-column) and score-per-list (standard deviation in brackets) for row and column players in each match-up.

row \ col	Kermit			Tracker			Tracker-rm36			theCount			theCount-rm36		
	r-c	row	col	r-c	row	col	r-c	row	col	r-c	row	col	r-c	row	col
Mistake0.03	379	874(46)	495(55)	344	812(47)	468(55)	262	756(48)	494(50)	108	756(48)	648(48)	133	761(48)	628(46)
Mistake0.10	-213	634(48)	847(48)	-90	693(48)	783(52)	47	684(48)	637(53)	-161	660(48)	821(48)	-144	662(49)	806(48)
Mistake0.25	-638	460(52)	1098(45)	-726	460(50)	1186(48)	-498	468(49)	966(56)	-618	511(50)	1129(52)	-564	493(52)	1057(53)
XSkat	-741	648(42)	1389(47)	-826	612(36)	1438(39)	-662	594(31)	1256(63)	-818	630(39)	1448(52)	-808	623(39)	1431(39)

between computer players. Our goal is to compare the performance of three different *candidate* players: Kermit, the Count and a player we call Tracker. Kermit is the static, non-adaptive skat player described in [Buro *et al.*, 2009]. The Count is a modification of Kermit that uses PIPMA to adjust its bidding behaviour, as described in Section 3. Tracker is a "naive" adaptive player also based on Kermit that tracks only its own performance as soloist in the context of its current opponent match-up: after each soloist game played, Tracker updates its perceived error in its bidding evaluation function according to Equation (1) from Section 3. It then uses the average accumulated error as an offset to its bidding threshold in future games. Tracker's final adjusted bidding threshold is capped at 0.75, so as to prevent the program from believing that all soloist games are unwinnable (and therefore never playing as soloist again to gather more data) after a particularly unlucky streak.

We note that the card-play strength of our three candidate players is identical. Furthermore, the adaptive bidding of Tracker and the Count is such that against an opponent of equal card-play strength, the bidding of these two programs is nearly identical to Kermit's. Therefore, comparing these programs by playing them against each other would be futile.

Instead, we play each of these candidates against a set of *benchmark players*. Three of these are synthetic players, as described in Section 3, using mistake rates of 0.03, 0.1 and 0.25. The fourth benchmark player is a rule-based program called XSkat, which is free software written by Gunter Gerhardt. XSkat has previously been shown to be substantially weaker than Kermit [Buro *et al.*, 2009], but serves as a useful example of a player that is very different from the synthetic players and our candidate players.

In our experiments, each candidate played the same set of 2000 skat deals against two clones of each of the benchmark players (although the candidate players were not aware that their two opponents were identical for the purpose of modelling). All experiments were run on computers with eight 2.5 GHz Intel Xeon E5420 CPUs with 16 GB of RAM.

We show results of these tournaments in Table 1. The displayed score represents average points earned per 36 games, which is the length of a standard match (or "list") in most skat clubs and tournaments. The candidate players Tracker and the Count have two types of entries in this table: one with the suffix "rm 36" and one without. The "rm 36" version of each program deletes all of its opponent modelling data every 36 games; the non-suffixed version keeps its data for the duration of the 2000 game run.

Discussion

In opponent modelling, the general goal is to improve performance against some opponents without losing in self-play or becoming vulnerable to exploitation by an adaptive adversary. Our experiments against the candidate players were designed to test the first two of these three criteria and we will discuss the results in that light.

Against the Mistake0.1 player, we see little difference between Kermit and the Count; in fact, Kermit performs slightly better. This is not unexpected, because Kermit's own global error rate is very close to 0.1, and therefore this result approximates a 'self-play' match. Since Kermit is tuned to perform well in self-play, the best the Count could hope for in this match-up is to tie Kermit's performance, which it effectively does.

Meanwhile, the Count achieves a modest but, in skat terms, still substantial gain against XSkat, and a dramatic gain against the exceptionally strong Mistake0.03 player. The Count's somewhat weak performance against the Mistake0.25 player was more of a surprise. Although the Count still outperforms Kermit in the long run, this advantage is slight enough that it disappears when the Count is forced to erase its modelling data. This appears to be because although the synthetic player's cardplay is quite weak, it still bids as aggressively as Kermit, meaning that Kermit earns many points on defense and would achieve little by increasing its own bidding aggressiveness (as the Count tries to do). XSkat, while of similar card-play strength to the Mistake0.25 player, is more conservative in its bidding. A follow-up experiment where the Mistake0.25 player's bidding threshold was increased from 0.6 to 0.7 seemed to confirm the role of the bidding behaviour; in this setting, the Count (with "rm36" enabled) exceeds Kermit's list-average by 57 points.

Finally, we compare the Count to the Tracker player. Our objective in including Tracker among our candidate players was to compare the amount of data needed to perform successful opponent modelling by a naive approach against the amount needed by the Count's PIPMA. In nearly all cases, we see that Tracker performs quite similarly to the Count in the long run. However, when it is forced to delete its modelling data every 36 games, the performance of Tracker degrades quickly, becoming weaker than both Kermit and the Count against all the benchmark players except Mistake0.03, where it is roughly even with Kermit. The Count, on the other hand, becomes only very slightly weaker when it deletes its data.

Effectively, Tracker is not able to accurately model its opponents in a short period of time using only end-of-game results whereas the Count gains considerably more information per game by examining the individual moves made by all players. This appears to allow the Count to model its opponents sufficiently quickly inside a 36 game window to make a positive difference in its play.

4.2 Online Play

In addition to the tournaments between computer players described above, the Count has been playing on ISS, an online skat server (`http://www.skatgame.net/iss/`). The population of this server consists of a few computer players (including Kermit and XSkat) as well as over 100 human players of varying playing strengths. Since being introduced to the server, the Count has played 14415 games and has achieved a list-average of 1000 (standard deviation of 18). The non-adaptive player Kermit had previously played over 280,000 games on this server, with a list average of 940 (standard deviation of 5). As Kermit is one of the top-ranked players on the server, a difference of 60 points is quite considerable, and strongly suggests that the Count's modelling approach works well against human players in addition to synthetic ones. As humans are typically good at exploitative play, this result also suggests that the Count's PIPMA approach is indeed robust against exploitation in practice, as suggested by our arguments in Section 3.

5 Conclusion

In this paper, we proposed Perfect Information Post-Mortem analysis as a simple tool for quickly assessing opponent decision-making abilities in certain classes of environments, notably trick-taking card games. We then used this technique to create the Count, a computer skat player that adjusts its bidding behaviour in real time in response to the card-playing strength of its opponents. We presented experimental results showing that the Count generally performs approximately as well or better — and in some cases, substantially better — as a non-adaptive player against several opponent types. Finally, the Count is able to successfully model its opponents in fewer than 36 games, the length of a standard skat tournament match, making it practical for real-time play against humans.

In pursuing the work we have described here, we have effectively been dealing with two separate questions: first, can we capture some model of the opponent's behaviour, and second, if so, what should we do about it. PIPMA is a relatively general technique that addresses the first of those questions in environments that are easier to deal with "in hindsight" than they were at the time that actual decisions were being made. However, PIPMA need not be restricted to measuring an opponent's global mistakes; the analysis could be partitioned in a number of different ways in an attempt to uncover much more specific opponent weaknesses. For example, in skat, one could distinguish between an opponent's mistakes as defender versus soloist, or mistakes made in early game stages versus those made later, and perhaps this information could be used to to exploit the opponent further.

The second question, how to appropriately react to a known opponent mistake rate, seems considerably more domain specific. In skat, we were fortunate in that a general approach to this question — adjusting bidding rate — seemed immediately apparent. However, although we performed various experiments to explore how the bidding rate should be adjusted, our final approach is still somewhat ad hoc and could likely be improved in future. In other domains, it is less apparent what the first steps should be in reacting to a known general opponent weakness.

Finally, we note that our focus here has been on opponent modelling in an adversarial setting. However, there are numerous non-adversarial contexts in which a PIPMA-style assessment of agent decision quality could be useful. Take, for instance, a card-game tutorial program that might wish to match a novice human player against computer players of an appropriate skill level. In fact, PIPMA may be even more easily applicable in such domains, since an administrating program can have access to full post-game hidden information even if the players involved normally would not. Thus, non-adversarial settings are another promising avenue along which this work could be further explored.

6 Acknowledgements

The authors would like to acknowledge GRAND NCE for their financial support.

References

[Billings *et al.*, 2002] Darse Billings, Aaron Davidson, Jonathan Schaeffer, and Duane Szafron. The Challenge of Poker. *Artificial Intelligence*, 134(1-2):201–240, 2002.

[Buro *et al.*, 2009] Michael Buro, Jeffrey R. Long, Timothy Furtak, and Nathan R. Sturtevant. Improving state evaluation, inference, and search in trick-based card games. In *IJCAI*, pages 1407–1413, 2009.

[Davidson *et al.*, 2000] A. Davidson, D. Billings, J. Schaeffer, and D. Szafron. Improved opponent modeling in poker. In *Proceedings of the 2000 International Conference on Artificial Intelligence*, pages 1467–1473. Citeseer, 2000.

[Gilpin and Sandholm, 2006] A. Gilpin and T. Sandholm. Finding equilibria in large sequential games of imperfect information. In *ACM Conference on Electronic Commerce*, 2006.

[Ginsberg, 2001] Matthew Ginsberg. GIB: Imperfect Information in a Computationally Challenging Game. *Journal of Artificial Intelligence Research*, pages 303–358, 2001.

[Johanson and Bowling, 2009] Michael Johanson and Michael Bowling. Data biased robust counter strategies. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 264–271, 2009.

[Richards and Amir, 2007] Mark Richards and Eyal Amir. Opponent modeling in scrabble. In Manuela M. Veloso, editor, *IJCAI*, pages 1482–1487, 2007.

[Southey *et al.*, 2005] Finnegan Southey, Michael Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. Bayes' bluff: Opponent modelling in poker. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 550–558, 2005.

[Zinkevich *et al.*, 2008] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret Minimization in Games with Incomplete Information. In *Advances in Neural Information Processing Systems 20*, pages 1729–1736, 2008.