Online Implicit Agent Modelling

Nolan Bard and Michael Johanson and Neil Burch and Michael Bowling
University of Alberta
Edmonton, Alberta
{nolan,johanson,burch,bowling}@cs.ualberta.ca

ABSTRACT

The traditional view of agent modelling is to infer the explicit parameters of another agent's strategy (*i.e.*, their probability of taking each action in each situation). Unfortunately, in complex domains with high dimensional strategy spaces, modelling every parameter often requires a prohibitive number of observations. Furthermore, given a model of such a strategy, computing a response strategy that is robust to modelling error may be impractical to compute online. Instead, we propose an implicit modelling framework where agents aim to estimate the utility of a fixed portfolio of pre-computed strategies. Using the domain of heads-up limit Texas hold'em poker, this work describes an end-to-end approach for building an implicit modelling agent. We compute robust response strategies, show how to select strategies for the portfolio, and apply existing variance reduction and online learning techniques to dynamically adapt the agent's strategy to its opponent. We validate the approach by showing that our implicit modelling agent would have won the heads-up limit opponent exploitation event in the 2011 Annual Computer Poker Competition.

Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*Games*

General Terms

Algorithms

Keywords

Economic paradigms::Game theory (cooperative and non-cooperative); Agent theories, Models and Architectures::Modeling other agents and self

1. INTRODUCTION

In any complex multiagent system the ideal agent behavior is contingent on the behavior of other agents. Hence, a key capability for agents in such domains is to learn about and adapt to other agents. Traditional agent modelling approaches observe an opponent's actions and construct a generative model of their behavior. This may involve directly estimating probabilities of actions, or fitting some parameters in a more complex model to describe how the agent's behavior is generated. We call this an *explicit model*.

Although explicit modelling provides a direct and rich representation of other agents' behavior, there are significant practical challenges. First, in complex systems, agent behavior is likely to be complex as well, and involve a very high-dimensional parameterization. Learning such an explicit model either requires significant prior knowledge, or a prohibitively large number of observations of their behavior. Second, even if we can estimate a sufficiently accurate explicit model, there remains the question of how to use the model to adapt our agent's behavior. The natural approach is to select actions that maximize the agent's utility given the estimated models, but this can be extremely vulnerable to model error as demonstrated by Johanson *et al.* [10]. They instead propose the offline computation of "robust responses" to models, but this computation is too slow to be used online for the complex systems of interest.

In this paper we propose *implicit modelling* of agents as a practical approach to adapting agent behavior for complex settings. Where explicit modelling uses online data to estimate a model and then determines its response, implicit modelling works backwards by first computing a portfolio of responses and then uses online data to estimate the utility of the responses in its portfolio. We call this implicit modelling as the other agents' behavior is summarized only by its effect on the utility of the agent's portfolio responses. This avoids the drawbacks of explicit models, as the computationally expensive robust response computations are performed offline. Furthermore, the dimensionality of the model parameterization is reduced to the size of the agent's portfolio, regardless of the complexity of the domain or agent behavior.

We begin by presenting the framework of extensive-form games, a general model of multiagent interaction. We then describe some related work and present our implicit modelling approach end-to-end: from building response strategies, to selecting responses for the portfolio, and finally using online learning algorithms to select the best strategy from the portfolio during online interaction. We empirically validate our approach using the domain of heads-up limit Texas hold'em, a complex domain where explicit modelling has generally been unsuccessful. We show that this approach outperforms other baseline approaches and that an agent using the framework would have won the 2011 Annual Computer Poker Competition's total bankroll competition, an event which highlights agents' abilities to model and adapt.

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.*

2. BACKGROUND

We begin by presenting the framework of extensive-form games as a model of multiagent interaction. We then discuss methods for computing behavior policies, called *strategies*, in this framework.

2.1 Extensive-Form Games

An extensive-form game is an intuitive model for representing interactions between multiple agents and their environment. In a repeated game such as poker, the agents play a series of independent games against each other, alternating positions (*i.e.*, being a different **player**) in each game. A single game is represented as a tree, in which each node represents a state where it is one of the players' turn to act (or chance's turn to act), and directed edges represent the available actions. Each leaf, called a **terminal node**, assigns a utility to each player. In games of imperfect information, each action may be observed by one or both players. When information is not observed, the players cannot determine the precise game state and instead only observe an **information set**, which is a set of game states indistinguishable to the acting player. For simplicity of exposition, our notation is presented for the situation of a two-player zero-sum game, *i.e.* where there is a single other agent in the game who will be called the opponent. However, the concepts can be equally applied in multiplayer and non-zero-sum settings.

A **strategy** for player $i \in \{1, 2\}$, σ_i, is a function mapping a player's information sets I to a probability distribution over the available actions, $A(I)$, and let Σ_i represent the set of such strategies. A **strategy profile** $\sigma = (\sigma_1, \sigma_2)$ is a tuple containing one strategy for each player in the game. σ_{-i} refers to the strategies in the profile σ for all players except player i. For a two player game, σ_{-i} just refers to the strategy of player i's only opponent. In domains such as poker where the agents alternate positions between iterations of the game, an **agent** itself is a strategy profile containing the strategies that are used when acting as each player. Given the strategies for two players, σ_1 and σ_2, we can define player i's **expected utility** as $u_i(\sigma_1, \sigma_2)$, which can be computed by traversing the game tree or approximated by Monte Carlo sampling of trajectories according to the strategies. Given an opponent's strategy σ_{-i}, we define the **best response value** for player i to be $b_i(\sigma_{-i}) = \max_{\sigma'_i \in \Sigma_i} u_i(\sigma'_i, \sigma_{-i})$, and **best response strategy** to be the maximizing strategy. A strategy profile is an **ϵ-Nash equilibrium** if the difference between each player's utility and the best response value to their opponent's strategy is less than or equal to ϵ. A **Nash equilibrium** occurs when $\epsilon = 0$ and each player is a best response to the other. Given an agent's strategy profile, we define **exploitability** to be the average of the best response values of its strategies, representing the expected loss against a worst-case adversary. A Nash equilibrium for two-player, zero-sum games has an exploitability of 0.

Poker is a canonical example of a stochastic, imperfect information, extensive-form game. In this paper we consider the variant called two-player limit Texas hold'em, which is the smallest variant played in the Annual Computer Poker Competition [8]. Two players play a long series of games against each other to win chips from each other, and the overall winner is the player with the most chips. Each game begins with each player being randomly dealt two cards that only they can see. The players then alternate turns wagering chips that their set of cards will be strongest at the end of the game. After the players have acted by betting or choosing to fold (surrendering the game), additional cards are revealed to all of the players to see and use, and the players place additional wagers. The game ends after four such rounds, at which time the remaining player with the strongest set of cards wins the wagered chips. While a Nash equilibrium strategy can be guaranteed to not lose against any adversary and thus do no worse than tie on expectation, ideal agent behaviour in the game comes from identifying and exploiting an opponent's weaknesses over the series of games to maximize the agent's total winnings.

2.2 Nash Equilibrium Approximation

CFR [16] is a state-of-the-art algorithm for approximating Nash equilibrium strategies in two-player zero-sum perfect recall extensive-form games. It is an iterative self-play algorithm. Each player begins with an arbitrary strategy. On each iteration, the players examine every decision, and for each possible action compare the observed value of their current policy to the value they could have achieved by making that action instead. This is the regret for playing an action, and the accumulated regret is used to determine the strategy used on the next iteration. In the limit, the average strategies used by the players converge to a Nash equilibrium.

Although CFR efficiently approximates a Nash equilibrium, human-scale problems are typically so large as to be beyond the capability of existing techniques using modern hardware. For example, two-player limit Texas hold'em poker has 3.19×10^{14} information sets, and solving the game with CFR would require 4.54 petabytes of RAM. The standard approach in such cases is to apply a state-space abstraction technique to derive a smaller **abstract game** that can be tractably solved [16, 6]. The resulting abstract strategy can then be used to select actions in the original game. The effectiveness of the resulting strategy depends on the abstraction technique used and the size of the abstract game; larger abstractions usually result in strategies that are less exploitable in the real game, although see the work of Waugh and colleagues [15].

2.3 Robust Counter-Strategies

An (approximate) Nash equilibrium strategy profile is guaranteed to not lose to a worst-case opponent. However, it will not obtain the maximum utility from a known, exploitable opponent. If the exploitable opponent's strategy is known, a best response lies at the other extreme from a Nash equilibrium: it is a **counter-strategy**, designed to maximally exploit an opponent's flaws. While a best response is the utility-maximizing strategy to use against the opponent, Johanson *et al.* have shown that such strategies are brittle. When used against a different opponent, or if the opponent's strategy changes or is estimated instead of known precisely, the best response strategy can lose badly [10]. An alternative approach is to use ϵ-safe best responses, as proposed by McCracken and Bowling [12]: the utility maximizing strategy from the set of strategies exploitable for no more than ϵ. In this setting, ϵ represents a maximum loss that we are willing to suffer in cases where our opponent modelling has failed. The Restricted Nash Response algorithm (RNR) is a technique for generating ϵ-safe best response strategies given a known adversary strategy [10]. A modified game is constructed which begins with a chance event only observed by player 2. With probability p, the result of this chance event

is that player 2 must play according to a fixed strategy $\sigma_2^{\mathbf{fix}}$, while with probability $1-p$ they may play the game as normal. Given a value of p, this modified game can then be solved using any game solving algorithm, such as CFR. In the Nash equilibrium of the modified game, player 1's strategy is an ϵ-safe best response to $\sigma_2^{\mathbf{fix}}$ for some value of ϵ. The role of player 1 and player 2 can be switched to get an ϵ-safe best response to $\sigma_1^{\mathbf{fix}}$. Varying p results in ϵ-safe best responses with different values for ϵ. This approach provides a strategy that limits its exploitability in the worst case, while still winning much more than a Nash equilibrium against the target opponent.

However, the RNR algorithm is only applicable when the opponent's strategy is known. A more realistic scenario occurs when we have a set of observations of the opponent playing the game and have to construct a model of their behaviour. If the model is incomplete or inaccurate, the RNR strategy may "overfit" to the opponent model and be ineffective against the actual opponent. Johanson *et al.*'s data biased response (DBR) algorithm [9] extends RNRs and performs well even when only observations of behavior are available. DBRs construct an opponent model by counting the frequency of taking each action at each information set over the set of observations. Instead of tuning a single probability p at the root of the tree as in RNR, a probability $p(I)$ is chosen at each information set I, with $p(I)$ scaling with the number of observations at I. The DBR strategy, as with RNR, is then trained using CFR. Setting and tuning an upper bound on $p(I)$, allows us to generate strategies with different degrees of exploitability ϵ, just as with RNR.

2.4 Related Work

Due to the intrinsic need for utility maximization in poker, it has been a common domain for agent modelling research with a focus on two-player limit Texas hold'em. Recent efforts to use explicit modelling in this domain suffer from the challenges of using explicit models during online interaction. Ganzfried and Sandholm attempt to learn an explicit model by combining strictly online observations with a prior (approximate equilibrium) strategy [4]. Though they show positive results against highly exploitable opponents, the effectiveness of the approach against strong opponents has not been explored. This explicit modelling based agent may be highly exploitable by strong opponents because it uses a best response to its current model, and the model must use a relatively coarse abstraction of the game for the agent to act quickly enough.

Ganzfried and Sandholm suggest that combining this approach with safe exploitation algorithms [5] may provide improved robustness. They demonstrate this combined approach in Kuhn poker (a small toy poker domain), but performance in large games has not been explored. With critical parts of their safe exploitation algorithms being either intractable without abstraction in large games (*e.g.*, computing Nash equilibria, or ϵ-safe strategies) or too computationally intensive for real-time computation (*e.g.*, exploitability), it is unclear how these techniques can scale to large domains.

Rubin and Watson compute the performance of different strategy adaptations against prior opponents offline, and use online observations to build a low-dimensional explicit model to evaluate similarity to prior opponents and identify relevant adaptations [14]. They show a marginal improvement over their non-adapting strategy, but the results are only for *expert imitators* of agents from the 2011 ACPC which may not be capable of capturing those agents accurately. Further, this technique has no guarantees on robustness as modelling error can lead to choosing a contraindicated adaptation.

Some prior work has examined agent modelling techniques with similarities to our implicit modelling framework. UCB1 [1] has been used to select from a portfolio of RNR strategies [10] or expert imitators [13]. Unfortunately, the results for portfolios of expert imitators were negative or statistically insignificant relative to a single expert. Further, these approaches ignore the fact that UCB1's regret bounds are for the stochastic bandit problem where there is no adversary who can manipulate the value of the bandit arms. As poker is an adversarial game, this may be inappropriate. This approach also ignores the potential value of off-policy estimation seen in non-stochastic bandit algorithms such as Exp4. Prior work by Hoehn *et al.* investigated techniques similar to Exp4 for selecting from a portfolio of strategies in Kuhn poker, but did not investigate how to scale this technique to larger domains [7]. Our implicit modelling framework, which we present next, addresses these issues and is demonstrated in two-player limit Texas hold'em.

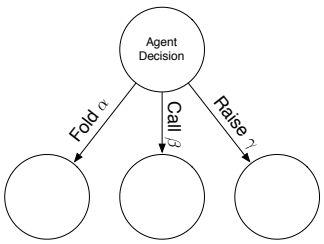
3. IMPLICIT MODELLING

Using explicit models in complex domains involving complex agent behavior presents two major challenges. First, the number of observations needed to learn a model generally depends on the degrees of freedom in the model. Without prior knowledge, the number of parameters for a general model must grow with the size of the agent's strategy space. For settings with many information sets, the number of observations required to learn such a general model will typically be prohibitively large, especially considering only a tiny fraction of information sets are observed with each playing. Second, even if we can learn a model quickly enough, best response strategies are brittle in the face of model error and computing robust responses in real-time is impractical for any non-trivial domains.

We propose to avoid these challenges altogether by avoiding the actual construction of an explicit model, in favor of an implicit model. Figure 1 shows graphically the differences between the two types of models. Generally, an explicit model represents particular action probabilities, (*e.g.* α, β and γ in Figure 1a). An implicit model instead summarizes these probabilities as the expected utilities of a portfolio of counter strategies (*e.g.* $u_i(\sigma_{-i}, \sigma_i^A)$ and $u_i(\sigma_{-i}, \sigma_i^B)$, the table entries in Figure 1b). Now consider increasing the complexity of the domain, such as increasing the number of information sets. The number of parameters in the explicit model grows with this increase in complexity, while the parameterization of the implicit model is unchanged, and remains the utilities for the fixed portfolio of responses.[1]

The implicit modelling approach comes with its own challenges. First, how do we choose the portfolio of strategies for modelling? Second, how do we estimate the utilities of the portfolio from online interaction, and how does the agent's behavior adapt given these estimates? In the sub-

[1]Formally, one can view the implicit model as a set of linear projections of the sequence-form representation of the opponents' joint strategy. And so one can view the implicit model as a particular low-dimensional representation of the agent's strategy. While an interesting observation, we don't use this fact further in this work.



(a) Explicit

	Portfolio		
	σ_i^A	σ_i^B	σ_i^C
σ_{-i}	10	0	-5

(b) Implicit

Figure 1: Explicit models versus implicit models.

sections that follow, we describe our end-to-end approach for addressing these challenges. First, though, observe that the explicit modelling approach gave little opportunity to do any useful computation offline, prior to the start of an interaction. In the implicit modelling approach, we have a more obvious offline/online computational distinction, which is to our advantage. Since our portfolio construction must occur prior to interaction, we are now free to use computationally demanding robust response approaches, such as RNR and DBR as discussed in section 2. By also limiting our actual behavior during play to be from the same portfolio of responses, we also get the advantage of having a "safety" guarantee that comes from using such ϵ-safe responses in our own behavior. We will see that this commitment also greatly simplifies the challenge of using the implicit model's estimates to adapt our agent's behavior, as we can leverage the existing online learning literature for this exact problem.

We give the details of our approach by first answering how to dynamically adapt our strategy online using a portfolio of responses as this choice impacts the portfolio's construction. We then consider the problem of constructing the portfolio of responses during the offline phase. An overview of the entire process of implicit modelling is presented in Figure 2.

3.1 Online Adaptation

With the portfolio computed offline, the online adaptation task becomes a utility estimation problem. If an oracle provided the expected utility for each of the portfolio's responses, we would simply act according to the response with highest expected utility. In lieu of the oracle, we can estimate the utility of each response using multi-armed bandit algorithms augmented by variance reduction techniques.

3.1.1 Variance Reduction

In stochastic games the actions of chance can significantly alter the game's final utility. The noise induced by chance can dramatically increase the number of observations necessary to have a confident estimate of the utility of a particular response. Variance reduction techniques can help eliminate some of this noise and reduce the number of observations needed to respond correctly.

For example, Bowling *et al.* used "imaginary observations" and importance sampling corrections in extensive-form games to provide a general technique for variance reduction for both on and off-policy utility estimation [3]. At a high level, this technique reduces estimator variance by imagining alternate observations that remain consistent with the actually observed actions taken by other agents (*e.g.*, alternate private cards or alternate actions that would end the game). Furthermore, this technique enables us to estimate off-policy, *i.e.* obtain estimates of the utility of one policy while behaving according to a different policy. Therefore, each observation of the opponents' choices and the outcome can be used to update estimates for the entire portfolio.

Unfortunately, using this technique off-policy has a caveat: the support of the acting strategy must be a superset of the strategies whose utility is being estimated. If not, the estimator can be biased. In other words, any sequence of actions that can be realized with non-zero probability (for some private information) for the off-policy strategy must be taken by the acting strategy with non-zero probability (with some other, not necessarily the same, private information). If this is not the case, then there exists some sequence of actions that the acting strategy will never take but that the off-policy strategy will, leading to bias.

One way to use Bowling *et al.*'s technique while avoiding this bias from off-policy **strategy incompatibility** is to sample according a strategy that mixes between all of the strategies in the portfolio. That way, if any individual strategy would play an action, then the mixture will necessarily play it with non-zero probability.

3.1.2 Bandit-Style Algorithms

Exp4 [2] is a well known algorithm for combining the advice of "expert" strategies to address the non-stochastic multi-armed bandit problem. For each time step of a multi-armed bandit problem, Exp4 generates a probability distribution over a collection of expert strategies using a normalized exponential function of the expected total rewards for each expert, selects an action according to the weighted mixture of the experts, receives a reward for the action, and computes the expected reward for each expert which is added to the vector of expected total rewards. The Exp4 action selection rule comes with finite-time regret bounds on the expected per-time-step regret, guaranteeing the expected utility of the selected actions performs nearly as well as the best expert in hindsight.

We can apply Exp4 directly to our task of choosing amongst the responses in our portfolio. However, we make two small changes. First, since we are mixing extensive-form strategies instead of a distribution over single actions, we must average the strategies' action sequence probabilities. Second, instead of uniform exploration over actions (or action sequences in our case) we force the weight of each expert to be at least some minimum value, bounding each expert's weight in the mixture away from zero. This allows us to guarantee the acting strategy has non-zero probability on every action sequence played by any response in the portfolio. With these modifications we can then use Bowling *et al.*'s off-policy importance sampling and imaginary observations to return low variance estimates of each response's utility and add it to the Exp4 accumulated total as normal.

3.2 Offline Portfolio Generation

One of the key benefits of the implicit modelling framework is that it allows us to build a portfolio of response strategies offline. Without the typically tight time constraints involved in real-time interaction, we have time to build more sophisticated responses for our portfolio. Ideally, our responses should be good at maximizing utility for the agents we will eventually interact with. Unfortunately, uncertainty about the other agent's strategies means that we also want robust strategies that are "safe" to use regard-

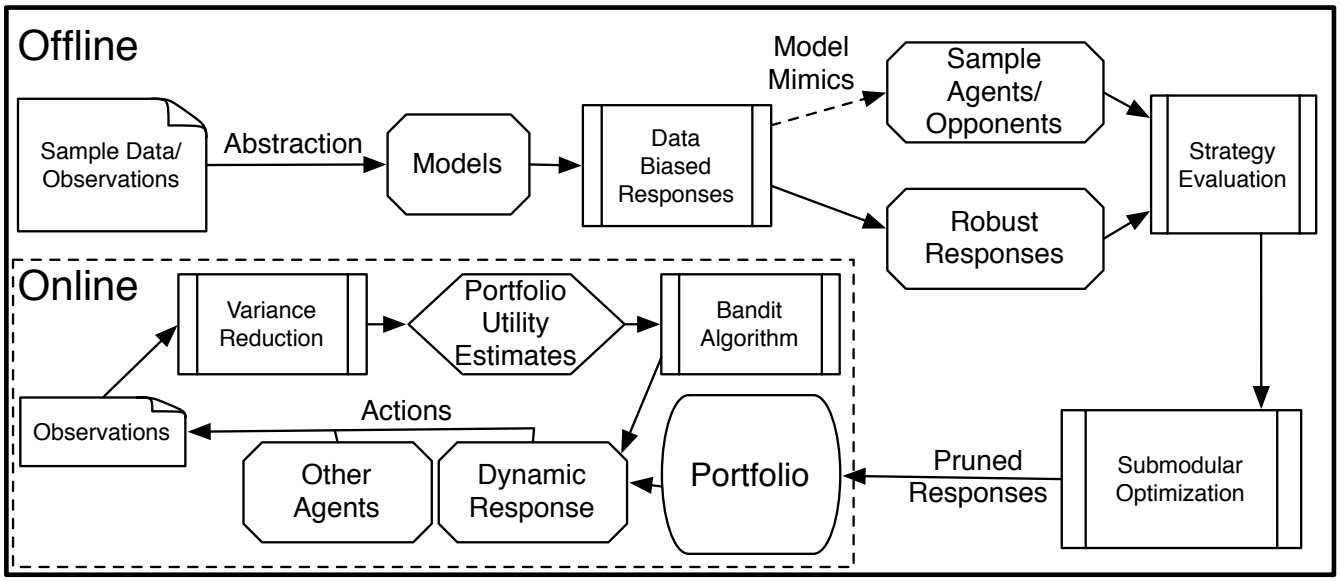


Figure 2: Implicit modelling process. The portfolio is constructed offline from robust responses to agent models and pruned with submodular optimization. It is then used online by a bandit algorithm in tandem with variance reduction techniques

less of the other agent's behaviour. In summary, we want a portfolio of strategies that can exploit a variety of other agents without any one strategy being badly exploitable.

3.2.1 Robust Responses

The goal of exploitive, yet robust, response strategies makes the restricted Nash response (RNR) and data biased response (DBR) techniques introduced in section 2 ideal tools for generating our portfolio of responses. In order to be able to use these tools, there must be some set of past interactions from which to construct responses. Interaction, in this case, is very loosely defined. There might be data from other agents playing the game, or even complete agent strategies available. Both of these might be publicly available (*e.g.* logs from an open competition), from the designer (*e.g.* a previously available non-modelling agent), or from the agents' own past matches. The choice of algorithm rests on the type of data: the RNR algorithm works best with a complete strategy, while the DBR technique is specifically designed to handle a collection of observed games.

We can also exploit an additional feature of the DBR algorithm. At the same time that it computes a robust response to the data, it also computes a robust strategy that mimics the data. At each information set, the **mimic** will, with some probability based on the amount of data available, choose its play so as to prevent exploitation by the DBR strategy. This mimic strategy behaves increasingly like the agent which produced the data as more observations are available. Note that there is no need for finding such a strategy when using an RNR with an explicit opponent model: the best possible mimic in such a case is the model itself, which is already available.

3.2.2 Pruning Responses

While we can generate a robust response from every past interaction, it may not be wise to include all such responses in the portfolio. An overly large portfolio has two drawbacks. First, it adds computational burden in order to estimate the utilities of every strategy in the portfolio after every hand. Second, and more importantly, both theoretical bounds and empirical practice of bandit-style algorithms (such as Exp4) show regret growing with the number of available bandit arms. Too many arms simply requires too much exploration before exploitation can reliably occur. Furthermore, each additional response may not be adding much to the overall exploitive power of the portfolio if other similar responses are already included. Finding a manageably-sized portfolio that still achieves broad exploitation possibilities is the final step in our approach.

Given a large set of responses computed from all past interactions, we want to find a subset of these responses which maximizes the resulting portfolio's exploitive power. This is a quintessential example of a submodular optimization [11]. Submodular optimization involves optimizing a function over sets, where the function exhibits diminishing returns, subject to a cardinality constraint on the set. In general, submodular optimization is NP-hard, however provably good approximations exist. In fact, greedily adding elements one at a time until reaching the capacity constraint, provides a $1 - 1/e$-approximation to the optimal subset.

Specifically, we would like to find some small subset of our entire set of generated responses, which retains as much of the exploitive power as possible. Because we cannot compute a given portfolio's exploitive power without interacting with other agents, nor do we know which agents we will eventually encounter, we will need a proxy objective for the portfolio's exploitive power. We define this proxy objective function on portfolios to be the expected utility the portfolio would obtain if an oracle told us which response from the portfolio to use when playing each of a given field of agents (*i.e.*, the response with the maximum expected utility for the agent at hand). This function is submodular, as adding strategies to the portfolio will eventually exhibit diminishing returns. Yet, we still need a field of agents to formally define our objective. This is where we can use our mimic strategies, which are generated as a by-product of DBR. Our objective

is then the total expected utility achieved against all of the generated mimic strategies if we can optimally choose the portfolio's utility-maximizing response for each mimic.

Using the greedy approximation, we repeatedly add responses to our portfolio one at time, with each one maximizing the marginal increase in our proxy objective function. If computational resources limit the number of responses then this can serve as our cardinality constraint and stopping condition. Alternatively, the marginal increase in value for including each additional response can provide a guide: stopping once the diminishing returns becomes too small.

4. RESULTS

Poker provides a natural domain for experimental validation of these techniques. Since human experts are known for quickly adapting to other players, this sets a high bar for agent modelling techniques to strive for. Furthermore, with a focus on statistical significance, seven prior competitions, and numerous competitors worldwide, the Annual Computer Poker Competition (ACPC) is the premier venue for computer poker research. In addition to the yearly competition, the ACPC also provides access to a benchmark server following the competition for competitors to evaluate against agents from that year's competition.

We validate our implicit modelling framework using the benchmark server from the 2011 ACPC. Specifically, we evaluate an implicit modelling agent in the competition's total bankroll event in heads-up limit Texas hold'em. The winner of this event is the agent with the highest expected winnings against all other competitors in the event. Although section 3 described the general approach for building implicit modelling agents, we first provide the details specific to our heads-up limit Texas hold'em agent.

4.1 Agent Design

With performance in the Annual Computer Poker Competition as our goal for these experiments, we chose to use data from past competitors to generate our responses. To keep training data and testing opponents separated, we used logs from the 2010 ACPC for generating our response strategies. This provided data for 13 different agents, each with a total of at least 8.4 million full information hands (*i.e.*, no missing card information) against other competitors from that year.

The data for each individual agent was used with Johanson *et al.*'s DBR technique [9] to compute a robust response. As we are dealing with a human-scale game, DBR requires an abstraction choice for both the opponent and the response strategy. For the opponent we chose a coarse abstraction where each round's chance outcome is bucketed into one of five abstract outcomes. With perfect recall of these abstract chance outcomes and no abstraction of player actions, this means there are 5, 25, 125, and 625 combinations of chance for each betting sequence on the preflop, flop, turn, and river betting rounds, respectively. For the responses, we used an abstraction somewhat smaller than the strongest entries in the actual 2011 competition. DBR also requires the specification of parameters that controls the trade-off between the response's exploitation and exploitability. We tuned these parameters to generate responses that were exploitable for approximately 100 mbb/h (milli big blinds per hand) by a best responding agent trained in their own abstract game. This threshold was kept relatively low to ensure that any of the responses could be used without substantial risk.

Our experimental results show the performance of our implicit modelling approach using two different portfolios: a portfolio of all of the responses (except the responses to our own 2010 submission to the ACPC), and a smaller four-response portfolio generated by the greedy approximation to the submodular optimization described in section 3.2.2.

We chose Exp4 parameter settings by performing experiments against our generated mimic strategies using the small portfolio. One of the best settings used a temperature parameter of $\eta = 0.025$ and a minimum probability of 2%, although the performance was largely insensitive to these parameter choices, with broad ranges of parameters giving similar results. After a smaller experiment verifying this choice of parameters using the large portfolio, we used these parameter settings in all subsequent experiments.

Johanson *et al.* explored the effects of various types of imaginary observations and value functions for strategy evaluation. For our experiments we create observations for all possible private cards and early folding opportunities. Early folds are provably unbiased, and although the all-cards technique can create bias under partial information (due to card-replacement effects and not knowing which cards the other agent holds) prior results suggest this bias is small while providing substantial variance reduction (*viz.*, [3], table 3).

4.2 Empirical Results

While the ultimate validation is comparing the approach in the context of the 2011 ACPC entrants, we begin our experimental validation in a number of simpler settings. The first is a comparison against the four mimics for which the small portfolio's responses were designed. This experiment gives us an idealized case where we avoid any concerns about being unable to respond to one of the opponents. Furthermore, because the opponents include two of the most exploitable agents from the 2010 ACPC, this experiment is an ideal scenario for exploitation. In all of the following stacked bar charts, the total bar height indicates the expected winnings against the field of opponents while the bar's composition specifies the proportion of the mean won by playing against each specified mimic. Note that the vertical order of the bar's components is the same as the order in the legend. Expected winnings are in milli big blinds per hand (mbb/h). The match length for all experiments was 3000 hands: the same length as the 2011 ACPC event. 95% confidence intervals on all of the expected winnings values are ±7mbb/h or smaller, except where noted. Essentially, visible differences in the graphs are statistically significant.

The results for this simple setting are in Figure 3. Our implicit modelling agent using the small portfolio (labelled Small-Portfolio) outperforms our range of other baselines. First, note that its winning rate is nearly double that of a fixed Nash equilibrium strategy, which used a considerably larger abstraction. Furthermore, the Small-Portfolio agent improves upon any individual response from the portfolio (ASVP, GS6_iro, LittleRock, longhorn) by at least 19.9%, demonstrating that it is able to tailor its strategy online to the opponent at hand. It also outperforms the Small-Static agent: a baseline also built using a DBR, but responding to a single aggregate model built with the same data used for the small portfolio's models. This suggests that the mimics are exploitable in at least partially independent ways and that modelling them as a single aggregate player harms the response's exploitive power.

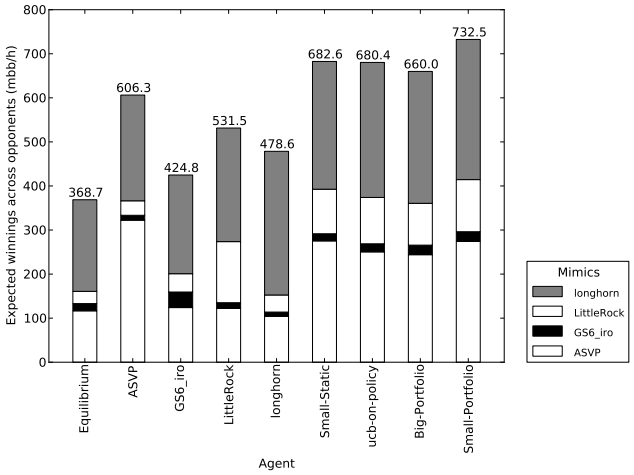


Figure 3: Performance versus mimics corresponding to the four responses in the small portfolio. Bar components are in the same order as the legend.

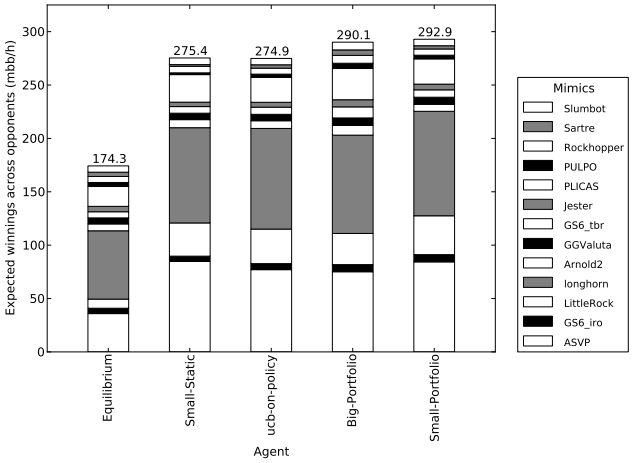


Figure 4: Performance versus mimics corresponding to the responses in the large portfolio. Bar components are in the same order as the legend.

Small-Portfolio also improves on an implicit modelling agent that uses upper confidence bounds (UCB) instead of our modified Exp4 algorithm to select from the portfolio. Note that this UCB based agent exhibited higher variance in its results, and a 95% confidence interval on its results are approximately ±13 mbb/h. Since UCB selects and acts according to a single strategy from the portfolio rather than a mixture, we cannot ensure that the support of the strategy being played is a superset of the supports for the portfolio's individual responses. Therefore we cannot estimate the utilities of the off-policy strategies in the portfolio without potentially introducing bias.

Finally, the smaller portfolio also outperformed the larger portfolio (Big-Portfolio). This demonstrates that even though Big-Portfolio contains a superset of the responses, there is a potential learning cost associated with including extraneous strategies in the portfolio.

Next, we evaluate the agents against the entire field of 2010 ACPC mimics. In contrast to the last experiment, the Small-Portfolio agent no longer has a response associated with every opponent, while the Big-Portfolio agent still does. The results are in Figure 4. Once again, the small portfolio agent outperforms the equilibrium, now by approximately 65%. Against this field, both Exp4-based implicit modelling agents outperform the aggregated Small-Static baseline agent and the UCB based implicit modelling agent. Despite the reduced number of responses, the Small-Portfolio agent's performance is within noise of the Big-Portfolio agent. These empirical results support our intuition for the benefits of using a submodular optimization to prune back the portfolio to a manageable size.

Finally, we validated the implicit modelling approach using the 2011 ACPC benchmark server. The benchmark server allows previous competitors to run matches against submissions from that year. Except for one agent that failed to run on the server, all other competitors were available. Table 1 shows the performance of the Big-Portfolio and Small-Portfolio agents from the previous two experiments contrasted with the competition's original results.

Our original submission to the competition in 2011 placed third with a total bankroll of 224 mbb/h with a 95% confidence interval of 6 mbb/h. Note that these results for the original competitors have been recomputed to exclude the agent that failed to run on the benchmark, although it did not change the outcome of the competition. To compute the results for our implicit modelling agents, data from our original agent "Hyperborean" (Hyperborean-2011-2p-limit-tbr) was excluded and new data was substituted for Hyperborean by playing matches on the benchmark server with our implicit modelling agents, Big-Portfolio and Small-Portfolio. The values for each agent's total bankroll in the modified competitions were recomputed using the data from these new matches combined with the remaining original data.

In this experiment both Big-Portfolio and Small-Portfolio do not have any responses tailored for the specific opponents since all responses were trained on data from 2010 ACPC agents. First, observe that both implicit modelling agents would have won the competition. Although the Big-Portfolio agent did not win by a statistically significant margin, Small-Portfolio wins the event by greater than the 95% confidence margin. These results demonstrate that the implicit modelling framework enables modelling and improved utility even against an unknown field of agents. Moreover, while the technique still outperforms other agents with a larger portfolio, pruning the portfolio of redundant or low value responses can improve performance further still.

5. CONCLUSION

We present an alternative approach to the traditional idea of modelling other agents with explicit models of their behavior. In complex domains, both learning a model of an agent and being able to respond robustly using a model can be impractical. Instead we present a framework for modelling agents implicitly through online estimates of the utility for a portfolio of robust responses computed offline. Through a synthesis of techniques from several areas including robust responses in extensive-form games, submodular optimization, variance reduction, and multi-armed bandits, we present a step-by-step approach of how to build implicit modelling agents. Using the domain of heads-up limit Texas

	2011 ACPC		Big-Portfolio		Small-Portfolio
Calamari	276 ± 4	Big-Portfolio	290 ± 11	Small-Portfolio	317 ± 5
Sartre	270 ± 6	Calamari	275 ± 5	Calamari	277 ± 4
Hyperborean	224 ± 6	Sartre	268 ± 8	Sartre	272 ± 6
Slumbot	214 ± 6	Feste	211 ± 6	Slumbot	216 ± 6
Feste	213 ± 5	Slumbot	209 ± 7	Feste	215 ± 5
ZBot	205 ± 6	Patience	207 ± 7	ZBot	207 ± 6
Patience	204 ± 6	ZBot	205 ± 8	Patience	205 ± 6
2Bot	187 ± 6	2Bot	188 ± 8	2Bot	187 ± 6
LittleRock	183 ± 6	LittleRock	182 ± 9	LittleRock	185 ± 6
GGValuta	147 ± 6	GGValuta	142 ± 7	GGValuta	147 ± 6
AAIMontybot	-31 ± 12	AAIMontybot	-28 ± 17	AAIMontybot	-33 ± 12
RobotBot	-36 ± 9	RobotBot	-42 ± 14	RobotBot	-41 ± 10
GBR	-54 ± 13	GBR	-58 ± 16	GBR	-61 ± 13
player.zeta	-189 ± 16	player.zeta	-205 ± 21	player.zeta	-203 ± 15
Calvin	-246 ± 12	Calvin	-243 ± 13	Calvin	-252 ± 12
Tiltnet	-287 ± 10	Tiltnet	-295 ± 13	Tiltnet	-300 ± 9
POMPEIA	-541 ± 6	POMPEIA	-548 ± 9	POMPEIA	-553 ± 6
TellBot	-738 ± 16	TellBot	-757 ± 17	TellBot	-783 ± 15

Table 1: 2011 ACPC total bankroll results for heads-up limit Texas hold'em. Results include those from original ACPC matches and matches played on the benchmark server against the Big-Portfolio and Small-Portfolio implicit modelling agents. 95% confidence intervals are shown and values are in milli big blinds per hand.

hold'em, we validate the approach through implementations of implicit modelling agents built using public data from the 2010 Annual Computer Poker Competition. These agents not only outperform other baseline response techniques, but also would have won the 2011 Annual Computer Poker Competition's exploitation event.

Acknowledgments

The authors would like to thank the members of the University of Alberta Computer Poker Research Group for helpful conversations pertaining to this research. This research was supported by NSERC, Alberta Innovates Technology Futures, and the use of computing resources provided by the 2011 Annual Computer Poker Competition's benchmark server, WestGrid, Calcul Quebec, and Compute Canada.

6. REFERENCES

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.

[2] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proc. of the 36th Annual Symposium on Foundations of Computer Science*, 1995.

[3] M. Bowling, M. Johanson, N. Burch, and D. Szafron. Strategy evaluation in extensive games with importance sampling. In *Proc. of the 25th Annual Int. Conf. on Machine Learning (ICML)*, 2008.

[4] S. Ganzfried and T. Sandholm. Game theory-based opponent modeling in large imperfect-information games. In *Proc. of the 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, 2011.

[5] S. Ganzfried and T. Sandholm. Safe opponent exploitation. In *Proc. of the 13th ACM Conf. on Electronic Commerce (EC)*, 2012.

[6] A. Gilpin, T. Sandholm, and T. B. Sørensen. Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of texas hold'em poker. In *Proc. of the 22nd National Conf. on Artificial Intelligence (AAAI)*. AAAI Press, 2007.

[7] B. Hoehn, F. Southey, R. C. Holte, and V. Bulitko. Effective short-term opponent exploitation in simplified poker. In *Proc. of the 20th National Conf. on Artificial Intelligence (AAAI)*, 2005.

[8] E. Jackson. The Annual Computer Poker Competition webpage. `http://www.computerpokercompetition.org/`, 2012.

[9] M. Johanson and M. Bowling. Data biased robust counter strategies. In *Proc. of the 12th Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2009.

[10] M. Johanson, M. Zinkevich, and M. Bowling. Computing robust counter-strategies. In *Proc. of the Annual Conf. on Neural Information Processing Systems (NIPS)*, 2007.

[11] A. Krause and D. Golovin. *Tractability: Practical Approaches to Hard Problems*, chapter Submodular Function Maximization. Cambridge University Press, 2012 (to appear).

[12] P. McCracken and M. Bowling. Safe strategies for agent modelling in games. In *AAAI Fall Symposium on Artificial Multi-agent Learning*, October 2004.

[13] J. Rubin and I. Watson. On combining decisions from multiple expert imitators for performance. In *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2011.

[14] J. Rubin and I. Watson. Opponent type adaptation for case-based strategies in adversarial games. In *Proc. of the 20th Int. Conf. on Case-Based Reasoning (ICCBR)*, 2012.

[15] K. Waugh, D. Schnizlein, M. Bowling, and D. Szafron. Abstraction pathologies in extensive games. In *Proc. of the 8th Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2009.

[16] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems 20 (NIPS)*, 2008.