Asymmetric Abstractions for Adversarial Settings

Nolan Bard
University of Alberta
Edmonton, Alberta
nolan@cs.ualberta.ca

Michael Johanson
University of Alberta
Edmonton, Alberta
johanson@cs.ualberta.ca

Michael Bowling
University of Alberta
Edmonton, Alberta
bowling@cs.ualberta.ca

ABSTRACT

In multiagent domains, an agent's beliefs about how other agents will or could act plays a significant role in their own behaviour. In large domains where it is infeasible to uniquely represent every possible decision an agent will face, abstraction is often used to collapse the state and action space to make the problem tractable. By abstracting other agents' views of the environment, the agent makes assumptions about how other agents act. Incorrect abstraction choices can yield less than ideal performance as other agents may, in reality, use coarser or finer abstraction than they were modelled with. The standard approach when abstracting is to use symmetric abstraction: where all agents are assumed to distinguish states in the same way. This work examines the benefits and potential pitfalls of using asymmetric abstractions in two-player zero-sum extensive-form games. Using the domain of two-player limit Texas hold'em poker, we investigate the performance of strategies using both symmetric and asymmetric abstractions in terms of in-game utility and worst-case utility in the real game. Furthermore, we show that combining asymmetric abstractions with robust counter-strategy techniques can produce counter-strategies which dominate their symmetric abstraction counterparts in terms of both exploitative power and worst-case utility.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

Game theory; extensive-form games; abstraction; agent modelling; opponent modelling; multiagent learning; poker

1. INTRODUCTION

In large decision-making scenarios it is common to use abstraction techniques to simplify the space of solutions and to make it tractable for our reasoning algorithms. The granularity of an abstraction presents a trade-off between computational requirements and the fidelity of the abstraction. In single-agent settings, one typically uses the finest granularity possible that is tractable given the available computation.

In multiagent settings, the situation is considerably more complicated. First, it is not only necessary to choose an abstraction for the agent of interest, but it is also necessary to choose an abstraction for the other agents in the environment. Such an abstraction choice is representing one's belief about the other agents' capabilities, knowledge, and computational capacity. In this sense, a fine-grained abstraction is not always the obvious choice. Further, given fixed computational resources, it is not obvious how to trade-off using those resources to have a finer-grained agent abstraction or a finer-grained abstraction for the other agents. The situation in multiagent scenarios is complicated by Waugh *et al.*'s abstraction pathologies [13], which show that in multiagent domains there is no guarantee that refining an abstraction will result in a better approximation of optimal behaviour.

Despite the lack of theoretical guarantees, there is considerable evidence that finer-grained abstractions do actually improve agent decision-making. For example, in Texas hold'em poker, finer-grained abstractions have been shown to perform better in head-to-head competitions [14, 4] as well as resulting in less exploitable behaviour in the worst case [8, 7]. However, all of this empirical evidence has been based on *symmetric* abstraction choices, where the same abstraction is used for all of the agents. There has been no empirical analysis in this game on the effect of asymmetric abstractions, *i.e.* different abstractions for the agents, on the quality of the resulting behaviour. As a result, there is no guidance to a practitioner for how they should trade off abstraction granularity between the agents. However, this is exactly one of the key choices faced by practitioners.

In this paper, we present the first thorough empirical exploration of asymmetric abstractions. We do this in the domain of two-player limit Texas hold'em, where others have shown the value of symmetric abstractions. We examine how the abstraction trade-off affects an agent's performance in both head-to-head competition and in terms of worst-case exploitability, which is often used as a measurement of Nash equilibrium approximation quality. Our results give the first guidance for practitioners on the abstraction trade-off and show that symmetric abstractions, while being the most common choice, may not be ideal. In addition, we explore the effect of abstraction choice when building counter-strategies from observations of other agents. In this case, there is an additional trade-off of representation capacity and sample efficiency.

2. BACKGROUND

We begin our exposition with an introduction of basic terms and concepts for extensive-form games, our experimental domain

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

of two-player limit Texas hold'em poker, and existing solution techniques for extensive-form games.

2.1 Extensive-Form Games

Extensive-form games are a natural formalism for describing the interaction of agents in an environment. These interactions are represented by a tree in which nodes represent game states at which some player acts, and edges represent the possible actions. Leaf nodes represent the end of the game, and assign utilities to the players. In games with random chance, there is a special *chance player* who acts randomly according to a *known* distribution. In games with imperfect information, some actions may not be observable by some of the players, who then cannot distinguish the exact state of the game. An **information set** is a set of game states that are indistinguishable to a player because of these unobserved actions. A player must act based solely on its current information set, thereby acting the same at all the indistinguishable game states within the information set.

A **behavioural strategy** (henceforth a **strategy**) is an explicit description of how a player will play a game: a mapping from information sets to probability distributions over the legal actions. Given the strategies for two players, σ_1 and σ_2, it is straightforward to calculate each player's expected utility by traversing the game tree, or approximating it through Monte Carlo sampling (*i.e.*, playing a large number of games). The field of game theory describes several types of strategies that an agent may want to calculate or approximate. First, a **best response** for player i to their opponent, $-i$, is the strategy σ_i' that maximizes their expected utility. One common measure of a strategy's quality is its expected loss when playing against its own best response, or **exploitability**. Although best responses maximize their expected utility against a particular opponent, they tend to themselves be highly exploitable. At the other extreme, a **Nash equilibrium** is a pair of strategies, σ_1^* and σ_2^*, that are mutually best responses to each other. In two-player zero-sum games (when alternating positions), the strategies in a Nash equilibrium are **unexploitable**: against any opponent, they can do no worse than tie on expectation. Computing or approximating such a strategy is called **solving** a game.

2.2 Poker

Poker is a canonical game of random chance and imperfect information. It is a repeated game in which a set of players play a long series of games, and aim to win as much as possible from their opponents. Poker is actually a family of games with similar rules; in this work, we will consider a variant called two-player (also known as *heads-up*) limit Texas hold'em, which is the longest-running event in the Annual Computer Poker Competition (ACPC) [1]. Each game begins with the players being forced to wager a small number of chips, and then progresses through four rounds. In each round the players are randomly dealt cards from a standard 52 card deck: two private cards that only they can see or use in the first round, and three, one, and one public cards on each subsequent round, respectively, that both players can see and use. After the cards are revealed in each round, the players use their chips to place wagers that their cards will be the strongest at the end of the game.

A player's goal in a poker game is to maximize their winnings against their opponent. While research has developed efficient techniques for approximating Nash equilibrium strategies, as we describe in Section 2.3, these strategies are concerned with worst-case opponents and do not attempt to exploit an opponent's errors. Equilibrium strategies usually win much less than a best response can against weak opponents. Alternatively, given some knowledge about our opponent's strategy, we can precompute an exploitative counter-strategy to win more against them than a Nash equilibrium would. We will discuss this approach in Section 2.4.

2.3 Nash Equilibrium Approximation

Counterfactual Regret Minimization, or **CFR** [14], is a state-of-the-art algorithm for approximating Nash equilibria in large two-player extensive-form games. CFR is an iterative self-play algorithm that simulates repeated games between two players. The players begin the game with arbitrary strategies σ_1 and σ_2. On each iteration of the algorithm, the players play against each other by traversing the game tree. At each decision, they compare the value of each action against the value of the current strategy. The difference in these values, called **regret**, is accumulated over all of the iterations of the algorithm, and the players update their strategies to choose their future actions proportional to the actions' positive regret. In the limit, the average strategy used by the players, $\bar{\sigma} = (\bar{\sigma}_1, \bar{\sigma}_2)$, converges to a Nash equilibrium. Although convergence to a Nash equilibrium is only guaranteed in the two-player, zero-sum, perfect recall setting, CFR has also been successfully applied to multiplayer games [11], non-zero-sum games [8], and imperfect recall games [7].

While CFR is memory and time efficient, even relatively small poker games played by human professionals remain too large to be solved using modern hardware. For example, the game of two-player limit Texas hold'em poker has 3.19×10^{14} information sets, while the largest known game solved using CFR has 3.8×10^{10} information sets [3].[1] The ubiquitous solution to this problem is to apply a **state-space abstraction** technique to the game. An abstraction can be represented as a many-to-one mapping from the information sets of the real game to information sets in a smaller, ideally strategically similar, **abstract game** that can be tractably solved. Applying a game solving algorithm such as CFR to the abstract game approximates an **abstract game Nash equilibrium**. While abstraction means that this strategy may not be optimal in the unabstracted game, it can be used to choose actions at real game information sets by applying the mapping and returning the strategy's chosen action. The success of this approach relies on how well the abstract game models the real game. The reduction in size necessary to produce a tractable abstract game typically requires at least some loss of information. If strategically dissimilar information sets are merged together, the resulting abstract strategy may not be effective in the real game, as measured by exploitability (worst-case loss) or in-game performance against other strategies.

Creating an abstraction typically involves two tasks. First, we must extract features, and choose a method for grouping information sets based on these features such as tiling the space [14, 4] or using a clustering algorithm such as k-means [7, 2]. Second, we must choose a number of abstract game information sets to which the game will be reduced. Increasing the size of the abstract game increases the time and memory required to solve the game, but also means that fewer information sets need to be mapped together which usually results in a better model of the game. While intuition would suggest that increasing the size of an abstraction will result in abstract game Nash equilibria that are less exploitable in the real game, there is no theoretical guarantee of this property. In fact, toy domains have yielded examples of **abstraction pathologies** where even strict refinements of an abstraction can result in increased exploitability [13]. In human-scale games, however, increasing the size of an abstraction

[1] Jackson's published size of 88 billion information sets includes terminal nodes. The 38 billion figure we list here is the usual count of only the decision points.

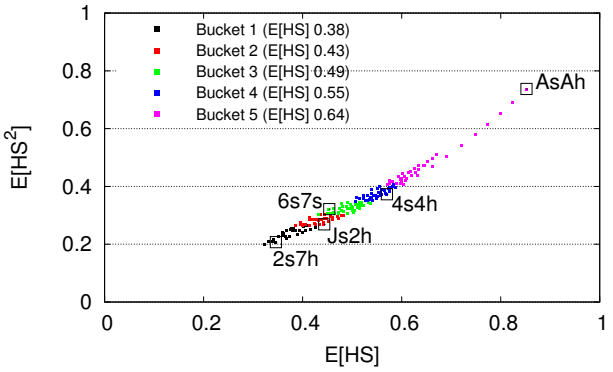

Figure 1: Abstraction of the first round of Texas hold'em poker, dividing 1326 hands into 5 percentile $E[HS^2]$ buckets.

Abstraction	Information Sets	CFR Memory
5-Bucket	3624290	140 MB
8-Bucket	23551424	934 MB
12-Bucket	118671936	4708 MB

Table 1: Sizes of percentile abstractions.

typically results in a consistent decrease of an abstract strategy's real game exploitability [8, 7].

In the limit Texas hold'em poker game we will use for our experiments, abstraction is applied only to the chance events in the game, and not to the players' actions. The abstraction task is thus simplified to finding similar sets of cards which are mapped together to form **buckets**. In our experiments, we will use two well-known abstraction features called **expected hand strength** ($E[HS]$) and **expected hand strength squared** ($E[HS^2]$) [14; 7, p. 5]. $E[HS]$ is a heuristic that assigns a value in [0,1] to each hand, representing the probability of winning against a randomly sampled opponent hand while taking the expectation over any unseen public cards. $E[HS^2]$ measures the expected square of hand strength, which promotes weak hands that might become strong, such as flush draws and straight draws. A **percentile** abstraction groups hands with similar feature values into approximately equal sized buckets. On the first round of the game, for example, a 5-bucket percentile $E[HS^2]$ abstraction groups the 1326 possible hands of cards into the buckets shown in Figure 1.

For each of the next three rounds, new public cards are revealed and each round's abstraction contains larger sets of cards that must be abstracted. This is done by taking each previous round's bucket, considering all ways to deal the new public cards to the hands in that bucket, and then reapplying the percentile $E[HS^2]$ technique to that set of hands. The abstractions we will investigate in this paper have the **perfect recall** property, in which hands that are mapped together on one round must also be mapped together on all earlier rounds. By convention, the same branching factor n is used on each round. In this paper, we will consider three standard sizes of abstractions used in prior research in abstraction techniques, with branching factors of 5, 8 and 12 buckets on each round. The 5 and 8 bucket abstractions will be partitioned according to percentile divisions of $E[HS^2]$, while the 12-bucket abstraction will first divide the hands into six $E[HS^2]$ sets, which are further split into two $E[HS]$ sets. The sizes of these abstract games and the amount of memory required by CFR to solve them is shown in Table 1.

2.4 Robust Counter-Strategies

Nash equilibrium strategies are useful in two player games because they minimize a player's exploitability. However, such strategies do not attempt to exploit an opponent's errors, and typically only win by a small margin against exploitable opponents. Since the goal of poker is to maximize winnings, a related line of research is to compute counter-strategies that exploit the errors of opponents or classes of opponents.

While a best response strategy is by definition a strategy that maximizes utility against a specific opponent, they make no attempt to limit their own exploitability and may lose badly when used against other opponents. **Robust counter-strategies** offer a compromise between exploiting an opponent and minimizing one's own exploitability. In particular, an **ϵ-safe best response** is a strategy from the set of strategies exploitable for no more than ϵ that maximizes utility against a particular opponent [10]. While best responses are then ϵ-safe for sufficiently high values of epsilon, we are more interested in ϵ-safe best responses that risk losing an acceptably small amount.

The **Restricted Nash Response (RNR)** algorithm [9] is an efficient way to compute ϵ-safe best responses. It takes as input a target opponent's strategy σ_{fix} and a parameter $p \in [0, 1]$ which trades off between minimizing exploitability and exploiting the opponent. We then use CFR or any other game solving algorithm to compute a Nash equilibrium for a modified game, where one "restricted" player is forced with probability p at the start of each game to play according to σ_{fix}, and with probability $(1 - p)$ is free to choose their own actions. We call these two parts of the restricted player's strategy their **model** and **response**, respectively. The other "unrestricted" player is always free to choose their actions. When we solve this modified game, the unrestricted player's strategy will converge to an ϵ-safe best response for some ϵ. Decreasing p results in less exploitable strategies, with an unexploitable Nash equilibrium occurring when $p = 0$. Increasing p results in strategies that exploit the opponent, with a best response occurring when $p = 1$. The Restricted Nash Response algorithm generates the Pareto-optimal set of trade-offs between these goals.

The Restricted Nash Response algorithm is difficult to use in practice, since we often do not have access to the opponent's strategy. Typically only observations of an opponent's behaviour are available when constructing a robust counter-strategy. In this setting, the related **Data Biased Response (DBR)** algorithm [6] allows us to create robust counter-strategies that exploit flaws demonstrated in the observations. First, we choose an abstraction to use for an opponent model. Next, we map the real game observations of our opponent into the abstract game and create our model from frequency counts of the observed actions. Finally, as with the RNR algorithm, we solve a modified game in which one player is sometimes forced to play according to the opponent model, and the other player converges to a robust counter-strategy. In DBR, the probability of being forced to follow the model is applied at each information set (as opposed to the start of the game), and varies with the number of times the opponent was observed at that information set, reflecting the model's varying accuracy throughout the game. By varying a single parameter $P_{\max}$, the maximum probability of following the model, we can produce a range of counter-strategies similar to RNR that trade off between minimizing exploitability and increasing exploitation. If too few observations are supplied to DBR, it fails gracefully by returning less exploitable strategies that are more similar to a Nash equilibrium. Increasing the number of observations allows us to generate counter-strategies that exploit more opponent errors while still limiting their worst-case loss.

3. ASYMMETRIC ABSTRACTIONS

For large multiagent domains where abstraction is needed to make the problem feasible, agent designers face the difficult task of choosing the abstractions which encode the players' knowledge of the real game. As in many domains, designers face a feature extraction problem where they must obtain features which provide discriminatory power between states where an agent should act differently. Unlike single agent domains, designers must also determine how to distribute their limited computational resources (typically limited by memory) between multiple agents.

To illustrate, consider the task of generating a strategy for a two-player zero-sum extensive-form game. A common approach to this is to approximate a Nash equilibrium using techniques like those introduced in Section 2.3. In this setting, designers must choose how to allocate their limited memory between the size of the abstraction for the strategy being created and the size of abstraction for the opponent's response strategy.

Prior research in poker has investigated or discussed abstraction techniques [14, 2, 7], but has focussed on the feature extraction problem and only provided analysis for symmetric abstractions where all agents use the same features. The only investigation of asymmetric abstractions where players view the game with different discriminatory power was performed by Waugh *et al.* in a toy poker domain [13]. The historical use of symmetric abstractions is both a default assumption and a matter of convenience, as a symmetric abstraction only requires a game to be solved once to compute all players' strategies, while an asymmetric abstraction requires us to solve the game once for each arrangement of the players' abstractions. Despite this prior work, there are unanswered questions about the practical uses of asymmetric abstractions in large domains. In particular, these abstraction choices may affect two common performance measures: one-on-one performance against other agents, and exploitability in the unabstracted game.

In terms of real game exploitability, Waugh *et al.*'s abstraction pathology results examined both symmetric and asymmetric abstractions in a toy poker game and showed that nothing can be guaranteed about worst-case performance in the real game if the opponent is abstracted [13]. While similar problems could occur in human-scale games such as Texas hold'em poker, a recent investigation by Johanson *et al.* has found that in practice, increasing the size of abstractions does result in a consistent decrease in exploitability [8]. Further, the recent development of a variant of the CFR algorithm called CFR-BR has made it possible to solve asymmetric abstract games where the opponent uses no abstraction [5]. This provably converges to an abstract strategy with the lowest possible real game exploitability. While these results provide valuable insight, they do not investigate the impact of abstraction size on asymmetric abstractions when both players are abstracted.

Although real game exploitability is an important objective measure of a strategy's quality, agents typically do not face their worst-case opponent and one-on-one performance against other agents may be more relevant in practice. Johanson *et al.*'s CFR-BR work highlighted this by noting that despite being optimal in terms of real game exploitability, CFR-BR strategies tended to perform worse one-on-one compared to strategies solved with CFR using a symmetric abstraction. This result suggests that there may be a trade-off between real game exploitability and one-on-one performance, but this trade-off has only been investigated in the extreme case where the opponent uses no abstraction. More interesting trade-offs may occur in other asymmetric abstractions when either our agent or the opponent's abstraction is relatively stronger.

Agent designers face even more unanswered questions when generating robust counter-strategies to agent models constructed from limited observations (as opposed to an agent's explicit strategy). In this case, even smaller domains which could otherwise be represented exactly may require designers to assume some form of generalization to prevent model sparsity (*i.e.*, insufficient sampling of agent behaviour across the model's possible decision points). This situation arises when using the DBR algorithm described in Section 2.4 as it uses state-space abstraction to provide this generalization. This further encumbers designers with selecting an abstraction for the model of the opponent in addition to abstractions for the robust counter-strategy being created and the opponent's unrestricted response strategy.

The prior work on robust counter-strategies [9, 6] explored only the default case of symmetric abstractions, and the possible advantages of using asymmetric abstractions have not yet been explored. Without this investigation, designers are left with a gap in guidance on how to select abstractions that will yield the best robust counter-strategies. In this work, we aim to guide agent developers by addressing these outstanding questions. We begin our analysis with an examination of asymmetric abstractions in the context of computing Nash equilibrium approximations and then provide further results regarding robust counter-strategy generation.

4. NASH EQUILIBRIUM APPROXIMATION

As discussed previously, the question of how asymmetric abstractions impact an agent's one-on-one performance and real game exploitability remains largely unaddressed. Our first experiment directly investigates this question for approximate Nash equilibria strategies.

Experimental Design. Nine approximate Nash equilibrium strategies were constructed using different pairs of abstractions for the strategy being created and the opponent's response strategy. Each player's strategy uses one of a 5, 8, or 12-bucket perfect recall percentile $E[HS^2]$ abstraction (see Table 1). The CFR algorithm described in Section 2.3 was used to generate a strategy for each pair of abstractions (3 symmetric and 6 asymmetric). For each of the three abstractions we also produced a strategy using the CFR-BR algorithm [5] which solves an asymmetric game in which the opponent uses no abstraction. Each strategy's exploitability was computed using the Accelerated Best Response algorithm [8], and the one-on-one expected value, measured in milli-big-blinds per game, between each pair of strategies was measured by playing 100 million duplicate games of poker (200 million games total). The size of each abstract game is listed, with asymmetric games requiring half of each abstraction's size from Table 1, and CFR-BR requiring 1096 megabytes of memory for the unabstracted opponent [5].

Empirical Results. Table 2 presents the results for these 12 strategies. Each strategy has a label "U-R" which indicates which abstraction was used for us and for the opponent's response. For example, "12-5" is a strategy using an asymmetric abstraction where our strategy uses the 12-bucket abstraction and assumes the opponent is using a 5-bucket abstraction. Strategies labelled "U-FULL" are solved using the CFR-BR algorithm in which the opponent plays the full (*i.e.*, unabstracted) game.

These results provide insight about several trends that arise when increasing abstraction sizes. First, examining the symmetric abstraction case (5-5, 8-8, 12-12), we see that as we increase abstraction size both mean utility against the field and exploitability improve. While Waugh *et al.*'s abstraction pathology results showed that this is not guaranteed by any theory, these results help explain why competitors in the Annual Computer Poker

	12-5	12-8	12-12	8-5	8-8	8-12	5-5	5-8	5-12	Mean	Exploitability	Size
12-5	0	-3	-6	20	18	16	43	41	41	18.970 ± 0.128	435.757	2424 MB
12-8	3	0	-3	23	22	20	36	35	35	18.890 ± 0.143	378.919	2821 MB
12-12	6	3	0	16	16	14	29	28	30	15.842 ± 0.175	289.227	4708 MB
8-5	-20	-23	-16	0	-3	2	22	21	24	0.662 ± 0.121	379.659	537 MB
8-8	-18	-22	-16	3	0	4	16	15	20	0.276 ± 0.144	312.762	934 MB
8-12	-16	-20	-14	-2	-4	0	12	12	16	-1.985 ± 0.099	255.845	2821 MB
5-5	-43	-36	-29	-22	-16	-12	0	3	7	-16.189 ± 0.112	317.1	140 MB
5-8	-41	-35	-28	-21	-15	-12	-3	0	5	-16.751 ± 0.153	283.37	537 MB
5-12	-41	-35	-30	-24	-20	-16	-7	-5	0	-19.714 ± 0.190	234.351	2424 MB
12-FULL	-22	-22	-21	-14	-13	-11	-2	-1	2	-11.526 ± 0.221	87.2765	3450 MB
8-FULL	-36	-36	-32	-26	-24	-21	-14	-12	-7	-23.093 ± 0.070	101.256	1563 MB
5-FULL	-54	-50	-45	-42	-38	-35	-29	-26	-21	-37.585 ± 0.150	122.385	1166 MB

Table 2: Cross table of approximate Nash equilibria strategies using various abstractions. The row player's expected value is shown in milli-big-blinds per game, rounded to the nearest integer. Values against individual opponents are computed through sampling 100 million duplicate hands (200 million hands total) and have 95% confidence intervals of at most 0.424. 95% confidence interval on the mean is shown.

Competition (ACPC) [1] endeavoured to produce progressively larger abstractions [12]. To explore the effects of increasing each player's abstraction independently, we must move to asymmetric abstractions.

Our first significant result in asymmetric abstractions is the discovery of the first abstraction pathologies outside of the toy poker game used by Waugh *et al*. Note that abstract game Nash equilibria in the 5-12 and 8-12 abstractions are both less exploitable than in the 12-12 abstraction: if our goal is to minimize our exploitability, we would do better by using a smaller abstraction for our own agent. Further, the 5-8 strategy which reduces the abstraction size for both players is also slightly less exploitable than 12-12. This counter-intuitive finding shows that abstraction pathologies are not only a problem in toy domains.

Examining the asymmetric abstractions where the opponent's abstraction is larger than our agent's, such as 5-8 or 8-FULL, we observe a trade-off between one-on-one performance and exploitability. In all cases, as the size of our opponent's abstraction increases (*i.e.*, we become more pessimistic about our adversarial opponent's abilities), our exploitability improves while our one-on-one mean utility decreases. The CFR-BR strategies (U-FULL) are the extreme case of this form of asymmetry and, as observed by Johanson *et al.* [5], pay a substantial cost in one-on-one utility in order to minimize exploitability. This result suggests that agent designers whose goal is to minimize worst-case performance should assume a pessimistic (*i.e.*, fine-grained) abstraction for other agents.

Finally, we investigate the asymmetric abstractions where our agent's abstraction is larger than the opponent's, such as 8-5 or 12-8. Here we see the opposite trend in the trade-off between one-one-one performance and exploitability: as we improve our agent's abstraction, our exploitability gets worse while our one-on-one performance improves not only in the mean utility, but also in the one-on-one utility against each individual strategy. This suggests that agent designers focused on the more practical one-on-one performance goal may want to increase the size of their agent's abstraction.

This investigation of Nash equilibrium approximations using asymmetric abstractions isolates two independent but related trends in how abstraction size affects agent performance. Although historically the standard approach has been to use symmetric abstractions, we have shown that this choice may be balanced but not optimal for either minimizing exploitability or maximizing one-on-one performance. In fact, these goals are at odds: the exploitability-minimizing 12-FULL strategy would lose in one-on-one play against the smallest 5-5 symmetric abstraction, while the performance-maximizing 12-5 strategy is also the most exploitable.

A designer's prior domain knowledge and their beliefs about the other agents in the environment will impact their priorities over these goals. For example, if worst-case outcomes corresponded to people being injured or killed, improving worst-case performance may be a designer's only goal. In this case, using a more pessimistic fine-grained opponent abstraction may improve worst-case performance by yielding strategies that guard against an opponent which better approximates the worst case (although abstraction pathologies prevent any guarantees). On the other hand, if a designer believes the other agents are unable or merely unlikely to act so as to produce worst-case performance, then choosing abstractions which optimize one-on-one performance, such as a fine-grained abstraction for the agent's strategy, may produce better results in practice. As we will demonstrate in the next section, these abstraction choices continue to be relevant when we create robust counter-strategies that strike a balance between these goals.

5. ROBUST COUNTER-STRATEGIES

Robust counter-strategies provide another approach to constructing agent strategies. Instead of trying to optimize performance assuming a best responding opponent, as with the previous approximate Nash equilibria approach, robust counter-strategy algorithms attempt to compromise between worst-case performance and exploiting knowledge of how other agents act. When this knowledge comes from observations of an agent rather than explicit knowledge of their strategy, designers usually need to transform the observations into a generative model of the agent's behaviour. In many domains, the number of information sets may make it infeasible for the model to represent each distinct information set. Furthermore, even if every information set can be represented, a limited quantity of observations can result in insufficient sampling to build an accurate model. This **model sparsity** is typically combated by agent designers generalizing about their observations in some form. DBRs address both the model representation and observation generalization problems by using state-space abstraction.

The prior work on robust counter-strategies only examined the techniques using small symmetric 5-bucket abstractions [6, 9] and does not tease apart the distinct roles of the opponent abstractions. While the abstraction for the opponent's response acts to prevent the resulting counter-strategy from becoming exploitable

by overfitting to the opponent model, the abstraction for the opponent model provides generalization across a limited set of agent observations. Without exploring asymmetric abstractions, we cannot answer how designers should select abstractions to produce the best robust counter-strategies from a given quantity of observations when using techniques based on state-space abstraction, like DBR.

Though not previously explored, note that both RNR and DBR can be used with asymmetric abstractions. In particular, recall that DBR generates robust counter-strategies using an opponent whose strategy is mixed at each information set between an unrestricted regret minimizing strategy and an opponent model [6, Equation 1]. These abstractions for the opponent's response and model do not need to be the same, and DBR can use a small abstraction for the model to ensure data density while using a large abstraction for the response to reduce the counter-strategy's exploitability. Likewise, the abstraction for our counter-strategy agent can be distinct from either of the opponent's two abstractions.

We will now directly investigate this aspect of creating robust counter-strategies with both symmetric and asymmetric abstractions. Though our results from Section 4 provide evidence for how designers should choose the abstractions for the agent's strategy and the opponent's response, an abstraction for the opponent model must also be chosen. Our experiments will directly examine how the quantity of observations and abstraction size used to build the opponent model impact DBR performance. Furthermore, we present results for RNRs and DBRs using both symmetric and asymmetric abstractions evaluated according to their true worst-case performance in the real game for the first time.

Experimental Design. To explore the impact of the abstraction size, we used a variety of different configuration parameters to generate several RNRs and DBRs based on knowledge of an exploitable opponent. RNRs trained using an agent's explicit strategy optimally trade between one-on-one and worst-case performance in their abstract game. In practice, it is uncommon to know an agent's strategy exactly and it must be inferred from observations. When building responses from observations, RNR is prone to an overfitting effect that DBR avoids, and so we compute DBR counter-strategies in such cases.

The opponent was created using CFR to solve a modified 8-bucket abstract game where payoffs were "tilted", as in the original presentation of DBR [6]. Specifically, the tilted opponent (falsely) believes that it will receive 25% additional utility when it wins the pot either through a showdown or the other player folding. When building DBRs, models of the opponent were created using $10^4, 10^5, 10^6$, or 10^7 full information hands of play (*i.e.*, private cards were revealed). Observations of the tilted opponent were gathered using the same "probe" agent as in [6] which never folds and calls or raises with equal probability.

Throughout these experiments we examine RNRs and DBRs generated using different combinations of the aforementioned 5, 8, and 12-bucket perfect recall percentile $E[HS^2]$ abstractions. Trend lines labelled "DBR-U-R-M" indicate that the corresponding strategies are DBRs generated using abstractions U, R, and M for the agent's robust counter-strategy, opponent's response, and opponent model, respectively. RNRs are only labelled "RNR-U-R" as their opponent model is the tilted opponent's actual 8-bucket strategy.

Finally, as described in Section 2.4 both RNR and DBR have parameters which control the counter-strategy's trade-off between one-on-one and worst-case performance. RNR uses the parameter p to specify the probability the opponent plays according to their model. DBR combines a parameter $P_{\max}$ with a function that maps a quantity of observations to a probability the opponent must play according to the opponent model. The original presentation of DBR found that the "0-10 linear" function (yielding $P_{\max}$ with 10 or more observations and a linear interpolation between 0 and $P_{\max}$ from 0 to 10 observations) to perform best in practice [6]. We used this 0-10 linear function and varied the value of $P_{\max}$ or p to create a range of strategies.

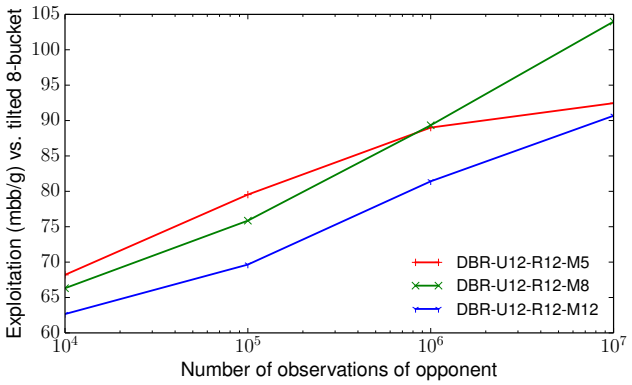

Figure 2: Impact of quantity of observations and model abstraction on exploitation of tilted 8-bucket equilibria. Exploitation values are in milli-big-blinds per game (mbb/g). Strategies for each data point are exploitable for 100 mbb/g in the 12-bucket abstract game.

As with our earlier Nash equilibrium results, strategies were evaluated in terms of exploitability and one-on-one expected utility against the tilted opponent. One-on-one expected utility was evaluated through sampling 100 million duplicate hands (200 million hands total). Values for one-on-one performance are in milli-big-blinds per game (mbb/g) and have 95% confidence intervals no larger than 0.585 mbb/g.

Empirical Results. Because DBRs are constructed from observations, the choice of abstraction for the opponent model directly impacts on our beliefs about the opponent. To isolate the interplay between the opponent model's abstraction and the quantity of observations, in our first experiment we fix the abstraction for the DBR strategy and the opponent's response strategy to the 12-bucket abstraction while varying the opponent model's abstraction size. Furthermore, we control each DBR's worst-case performance by computing DBRs that are exploitable for 100 milli-big-blinds per game in the 12-bucket abstract game. Figure 2 shows the one-on-one performance trends for DBRs using each of the three abstractions sizes for the opponent model as we vary the quantity of observations used to create the model.

These results demonstrate that using asymmetric abstractions for the opponent model and the opponent's response can produce strictly better robust counter-strategies. With fewer observations (10^4 and 10^5) we would improve our one-on-one performance against the tilted opponent while keeping the robust counter-strategy's worst-case performance in the abstract game the same. As the number of observations increases the DBRs using the 8-bucket opponent model eventually catch up (10^6) and then surpass (10^7) the DBRs using the 5-bucket opponent model. This crossover point is likely due to the small number of observations being too sparsely spread across the 8-bucket opponent model. Despite incorrectly representing the tilted opponent's true abstraction (8-buckets), the coarser 5-bucket abstraction yields better generalization and a more functional opponent model than the 8-bucket abstraction with very few observations. With a larger number of observations, the 8-bucket opponent model can better populate and

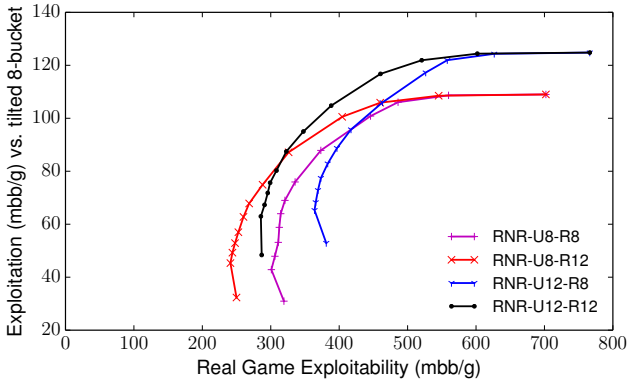

Figure 3: Impact of counter-strategy and opponent response abstraction size on RNR one-on-one and worst-case performance in the unabstracted game. Values are in milli-big-blinds per game (mbb/g).

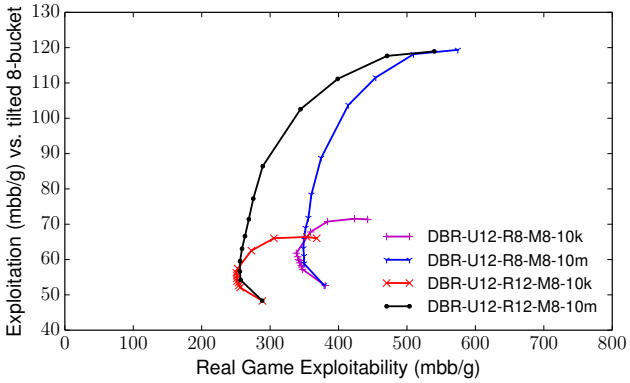

Figure 4: Impact of opponent response abstraction size and quantity of observations on DBR one-on-one and worst-case performance in the unabstracted game. Values are in milli-big-blinds per game (mbb/g).

capitalize on its correctly chosen model whereas the 5-bucket abstraction remains an incorrect model of the tilted opponent. We also observe that the 12-bucket abstraction does poorly throughout. This is unsurprising as it would needlessly separate observations from the tilted opponent's underlying 8-bucket strategy into the larger 12-bucket opponent model, resulting in both greater model sparsity and an incorrect representation of the opponent's abstraction. This suggests that designers should aim to select opponent model abstractions that are not only similar to their opponent's true abstraction, but can also be estimated accurately from their limited observations.

Prior work on robust counter-strategies evaluated the RNR and DBR techniques in terms of worst-case performance in the abstract game. Computing the real game exploitability of strategies has only recently become feasible [8], and as a result we can now revisit these techniques and evaluate their counter-strategies' true exploitability for the first time.

Our results in Section 4 suggested that increasing the size of our agent's and the opponent's response abstraction both traded between one-on-one and worst-case performance in contrary ways. Using RNRs and DBRs in different abstractions, we investigate if these trends also hold for robust counter-strategies. Performance curves were generated by computing a counter-strategy for each of the following p and $P_{\max}$ parameters: 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99, 1.0. We begin by examining the performance of RNRs in Figure 3 as this eliminates the impact of limited observations.

Examining the best values for one-on-one and worst-case performance, we see similar trends to those observed with approximate Nash equilibria. Comparing RNRs using an 12-bucket instead of an 8-bucket abstraction for the agent (RNR-U8-R8 to RNR-U12-R8, and RNR-U8-R12 to RNR-U12-R12), we see that the curves move up and to the right corresponding to improved one-on-one and poorer worst-case performance. Interestingly though, the RNRs using the larger 12-bucket opponent abstractions not only improve in worst-case performance relative to the RNRs using the smaller 8-bucket opponent abstractions (RNR-U8-R8 to RNR-U8-R12, and RNR-U12-R8 to RNR-U12-R12), but their performance curves also *dominate* them.

Note that since RNRs use the opponent's exact strategy, at p of 1 they become a best response to the opponent model and are oblivious to the abstraction chosen for the opponent's response. On the other hand, DBRs with a $P_{\max}$ of 1 only *approach* a best response in the limit of observations. Therefore, the opponent's response abstraction will always have some, though diminishing, impact on DBRs even with $P_{\max}$ set to 1. Figure 4 illustrates this difference, showing that DBRs, depending on the quantity of observations used, behave between the RNR domination and the more direct trade-off between one-on-one and worst-case performance observed with Nash equilibrium approximation. Comparing the DBRs using 10^7 observations (DBR-U12-R8-M8-10m and DBR-U12-R12-M8-10m), we see tends similar to the RNRs in Figure 3: using a larger opponent response abstraction produces nearly dominant performance curves. Unlike RNRs though, we can observe that using the larger 12-bucket opponent response abstraction results in a slight loss in the best one-on-one performance possible. For DBRs using only 10^4 observations (DBR-U12-R8-M8-10k and DBR-U12-R12-M8-10k) a more distinct trade-off between one-on-one and worst-case performance is apparent. With fewer observations, the DBRs using the larger 12-bucket opponent response abstraction have performance curves which shift distinctly down and to the left, corresponding to improved exploitability and poorer one-on-one performance. Both the RNR and DBR results echo the findings of Section 4: larger abstractions for our agent's abstraction should be used for greatest possible one-on-one performance, while larger opponent abstractions can produce better worst-case exploitability.

Finally, Figure 5 shows how varying the opponent model's abstraction affects one-on-one and worst-case performance in the real game. We use the same abstractions as in Figure 2, but create DBRs trained with 10^5 observations. RNRs using the 12-bucket abstraction for the robust counter-strategy and the opponent's response are also shown. Performance curves were generated by computing a counter-strategy for each of the following p and $P_{\max}$ parameters: 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99, 1.0.

Surprisingly, we observe that the robust counter-strategies with positive weight on the opponent model can be *less exploitable* than abstract game Nash equilibria (*i.e.*, RNR or DBR with p or $P_{\max}$ of 0) using the same abstraction. Figure 3 demonstrates this improved exploitability with RNRs while Figures 4 and 5 show this improvement is even more dramatic for DBRs. This may allow us to construct robust counter-strategies at no cost to worst-case performance relative to an abstract game Nash equilibrium. That

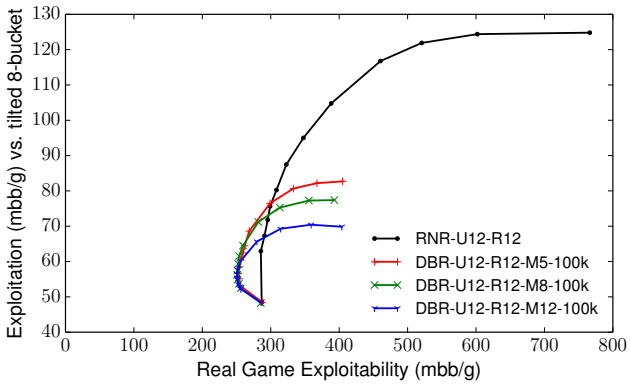

Figure 5: Impact of opponent model abstraction size on one-on-one and worst-case performance in the unabstracted game with 100,000 observations. Values are in milli-big-blinds per game (mbb/g).

said, these results may be specific to the particular tilted opponent we used. Furthermore, these robust counter-strategies still do not, and could not, have better exploitability than the 12-FULL CFR-BR strategy from Table 2. Although the cause of this effect is currently unknown, it appears similar to an effect discovered by Johanson *et al.* in their investigation of tiled equilibria, in which over-aggressive abstract strategies were found to be less exploitable than Nash equilibria within the same abstraction [8, Table 3].

6. CONCLUSION

In large multiagent domains where some form of state-space abstraction is necessary to make the problem tractable, agent designers are often faced with the difficult task of dividing limited computational resources for representing each agent's behaviour between the agents in the environment. While the standard approach in the poker domain is to use a symmetric abstraction that divides the resources evenly between the players, we have shown that this choice does not optimize for either worst-case performance or performance against suboptimal opponents. Our experiments performed the first empirical analysis of asymmetric abstractions in the human-scale game of two-player limit Texas hold'em, and addressed both Nash equilibria and counter-strategies. In the equilibrium approximation setting, we discovered the first abstraction pathologies outside of a toy domain. In the counter-strategy setting, we found that in addition to choosing abstractions for both players, the size of the abstraction used to model the opponent can be chosen to match the quantity of data and result in higher overall performance. Finally, we performed the first experiments with robust counter-strategies that measured real game exploitability, and found that robust counter-strategies can occasionally be less exploitable than abstract game Nash equilibrium strategies.

Acknowledgements

The authors would like to thank all of the members of the Computer Poker Research Group at the University of Alberta 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 Westgrid, Calcul Québec, and Compute Canada.

7. REFERENCES

[1] The Annual Computer Poker Competition webpage. `www.computerpokercompetition.org`, 2012.

[2] 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 *Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI-07)*. AAAI Press, 2007.

[3] E. Jackson. Slumbot: An implementation of counterfactual regret minimization on commodity hardware. In *2012 Computer Poker Symposium*, 2012.

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

[5] M. Johanson, N. Bard, N. Burch, and M. Bowling. Finding optimal abstract strategies in extensive-form games. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12)*, 2012.

[6] M. Johanson and M. Bowling. Data biased robust counter strategies. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS-09)*, 2009.

[7] M. Johanson, N. Burch, R. Valenzano, and M. Bowling. Evaluating state-space abstractions in extensive-form games. In *Proceedings of the Twelfth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-13)*, 2013.

[8] M. Johanson, K. Waugh, M. Bowling, and M. Zinkevich. Accelerating best response calculation in large extensive games. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI-11)*, 2011.

[9] M. Johanson, M. Zinkevich, and M. Bowling. Computing robust counter-strategies. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS-07)*, 2008.

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

[11] N. A. Risk and D. Szafron. Using counterfactual regret minimization to create competitive multiplayer poker agents. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-10)*, 2010.

[12] T. Sandholm. The state of solving large incomplete-information games, and application to poker. *AI Magazine*, Special issue on Algorithmic Game Theory, Winter:13–32, 2010.

[13] K. Waugh, D. Schnizlein, M. Bowling, and D. Szafron. Abstraction pathologies in extensive games. In *Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-09)*, pages 781–788, 2009.

[14] 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-07)*, 2007.