Playing Repeated Stackelberg Games with Unknown Opponents

Janusz Marecki
IBM T.J.Watson Research
P.O. Box 218
Yorktown Heights, NY 10598
marecki@us.ibm.com

Gerry Tesauro
IBM T.J.Watson Research
P.O. Box 218
Yorktown Heights, NY 10598
gtesauro@us.ibm.com

Richard Segal
IBM T.J.Watson Research
P.O. Box 218
Yorktown Heights, NY 10598
rsegal@us.ibm.com

ABSTRACT

In Stackelberg games, a "leader" player first chooses a mixed strategy to commit to, then a "follower" player responds based on the observed leader strategy. Notable strides have been made in scaling up the algorithms for such games, but the problem of finding optimal leader strategies spanning multiple rounds of the game, with a Bayesian prior over unknown follower preferences, has been left unaddressed. Towards remedying this shortcoming we propose a first-of-a-kind tractable method to compute an optimal plan of leader actions in a repeated game against an unknown follower, assuming that the follower plays myopic best-response in every round. Our approach combines Monte Carlo Tree Search, dealing with leader exploration/exploitation tradeoffs, with a novel technique for the identification and pruning of dominated leader strategies. The method provably finds asymptotically optimal solutions and scales up to real world security games spanning double-digit number of rounds.

Categories and Subject Descriptors

G [**3**]: Probabilistic algorithms (including Monte Carlo)

General Terms

Algorithms

Keywords

Stackelberg Games, Monte-Carlo Tree Search

1. INTRODUCTION

Recent years have seen a rise in interest in applying game theoretic models to real world security domains, ranging from allocation of security checkpoints at Los Angeles International Airport [13] to the analysis and detection of computer network intrusions [1, 10]. As these security domains impose non-simultaneous player actions, they are naturally modeled as Stackelberg games [4] wherein one player (referred to as the leader) commits to a mixed strategy of its choice, while the second player (referred to as the follower) responds based on the observed leader strategy. This type of approach has received a lot of attention, resulting in new methods for the scale-up of the proposed algorithms to games with large number of follower types [12] or large leader strategy spaces [6].

In arriving at the optimal leader strategies for such games, of critical importance is the leader's ability to profile the follower [14]. In essence, determining the preferences of the follower actions is a necessary step in predicting the follower best responses to leader actions, which in turn are necessary for finding the optimal leader strategy. If these follower preferences cannot be determined exactly, one can consider Stackelberg formulations with distributional uncertainty over the follower payoffs [14]. However, the best leader strategies in such games are often conservative [7] and prior work only considered the case of single-round games. The repeated-game Stackelberg scenario, wherein the leader can exploit extra information in the form of the follower responses, was not studied. A notable advance for repeated games was recently reported in [9], wherein the authors develop an elegant method for choosing leader strategies to uncover the follower preferences in as few rounds as possible. To our knowledge, however, no attempts have been made to *exploit* the revealed information about follower preferences to optimize total leader payoff over the rounds of the game.

This paper remedies these shortcomings by providing a first-of-a-kind method to balance the exploration of the follower payoff structure versus the exploitation of this knowledge to optimize on-the-fly the expected cumulative reward-to-go of the leader. By coupling Monte-Carlo Tree Search sampling to estimate the utility of leader mixed strategies with preemptive pruning of dominated leader strategies, we show how to effectively handle a broad class of repeated Stackelberg games often employed to model real world domains. We first provide a brief formal description of the decision problems at hand and recall the Bayesian Stackelberg game model. We then develop our algorithm and a separate method for pruning of dominated leader strategies. We finally provide an empirical evaluation of our method and discuss our results in the context of related work.

2. PROBLEM STATEMENT

2.1 Bayesian Stackelberg Games

A Bayesian Stackelberg game assumes a leader agent of a single type and a follower agent of type drawn from a set Θ. The set of pure strategies of the leader is $A_l = \{a_{l_1}, ..., a_{l_M}\}$ and the set of pure strategies of the follower is $A_f = \{a_{f_1}, ..., a_{f_N}\}$. Game payoffs are described in terms of the player utility functions: Leader's utility function is

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

$u_l : A_l \times A_f \rightarrow \mathbb{R}$ while the follower utility function $u_f : \Theta \times A_l \times A_f \rightarrow \mathbb{R}$ is unique for each type $\theta \in \Theta$ of the follower. The leader acts first by committing to a mixed strategy $\sigma \in \Sigma$ where $\sigma(a_l)$ is the probability of the leader executing its pure strategy $a_l \in A_l$. (Mixed strategies allow for higher expected payoffs of the leader as shown in [12].) For a given leader strategy σ and a follower of type $\theta \in \Theta$, the follower's best response $B(\theta, \sigma) \in A_f$ to σ is a pure strategy $B(\theta, \sigma) \in A_f$ that satisfies:

$$B(\theta, \sigma) = \arg \max_{a_f \in A_f} \sum_{a_l \in A_l} \sigma(a_l) u_f(\theta, a_l, a_f). \quad (1)$$

Given the follower type $\theta \in \Theta$, the expected utility of the leader strategy σ is therefore given by:

$$U(\theta, \sigma) = \sum_{a_l \in A_l} \sigma(a_l) u_l(a_l, B(\theta, \sigma)). \quad (2)$$

Given a probability distribution $P(\theta)$ over the follower types, the expected utility of the leader strategy σ over all the follower types is hence:

$$U(\sigma) = \sum_{\theta \in \Theta} P(\theta) \sum_{a_l \in A_l} \sigma(a_l) u_l(a_l, B(\theta, \sigma)). \quad (3)$$

Solving a single-round Bayesian Stackelberg game involves finding $\sigma^* = \arg\max_{\sigma \in \Sigma} U(\sigma)$.

An example Stackelberg game is depicted in Figure 1. Here, the leader agent (the security force) first commits to a mixed strategy. The follower agent (the adversary) of just a single type then observes the leader strategy and responds optimally to it, with a pure strategy, to maximize its own payoff. For example, the leader mixed strategy to "Patrol Terminal #1" (abbr. "PT1") with probability 0.5 and "Patrol Terminal #2" with probability 0.5 provokes the follower response "Attack Terminal #1" (abbr. "AT1"), because it provides the follower with the expected utility of $0.5 \cdot (-2) + 0.5 \cdot (2) = 0$ which is greater than the expected utility of $0.5 \cdot (2) + 0.5 \cdot (-4) = -1$ if the follower were to "Attack Terminal #2". One can calculate that the best leader mixed strategy for the Stackelberg game in Figure 1 is a pair [PT1=60%, PT2=40%] which (assuming that the follower breaks ties in the leader's favor) provokes the follower response AT1 thus providing the leader with the expected utility of $EU([PT1 = 60\%, PT2 = 40\%]) = 0.6 * 6 + 0.4 * 3 = 4.8$. (Note, how this mixed strategy is superior to the leader pure strategies [PT1=100%, PT2=0%] and [PT1= 0%, PT2=100%] which illustrates the benefits of randomized strategies in security domains.)

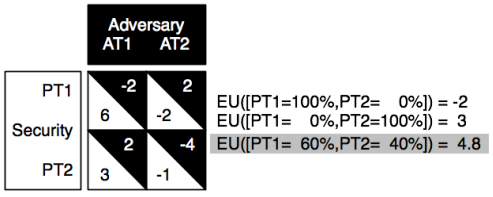

Figure 1: Single round Stackelberg game

2.2 Repeated Game Formulation

We adopt the model of repeated Bayesian Stackelberg games [9] which assumes that nature draws a follower of type $\theta \in \Theta$ at the start of the game, and then the leader plays H rounds of a Stackelberg game against a follower with fixed type θ. The formulation also posits that the follower plays a myopic (non-strategic) best-response strategy to the leader strategy observed in each round. (We defer to future work the more general case where the follower may also behave strategically, and may utilize a distribution over unknown leader preferences.) As such, the leader may never know the actual follower type θ that it is playing against, but it can infer the parts of the opponent payoff structure by observing follower responses to various leader actions. Whereas the objective in [9] was to minimize the number of rounds needed to exactly identify the follower type θ, our objective is to compute the best leader strategy in each round, given $P(\theta)$ and history of play in prior rounds, to maximize total leader payoff over H rounds of play.

To illustrate this concept refer to a two-round Stackelberg game in Figure 2 where the follower payoffs are initially unknown (uniformly distributed), represented by missing values in Table (a). Suppose the leader action in the first round is [PT1=100%,PT2=0%]. If the follower responds with action AT1 (refer to Table (b)), the leader receives a first-round payoff of 6, and infers that $u_{11} > u_{12}$. Consequently, in the second round the leader will again play [PT1=100%,PT2=0%], for it assuredly provokes the follower response AT1 and hence provides the leader with another payoff of 6, for a total two-round payoff of 12. (Note, that as the leader strategy [PT1=100%,PT2=0%] will not provide additional information about the follower payoffs, it will not be chosen in the second round of the game by the algorithm introduced in [9].) On the other hand, if the follower replies in the first round to [PT1=100%,PT2=0%] by playing AT2 (refer to Table (c)), the leader receives first-round payoff of -2, and infers that $u_{11} < u_{12}$. In that case, the optimal leader strategy in the *second* round is [PT1=0%,PT2=100%] yielding an expected payoff of $0.5 \cdot (3 + 1) = 2$ in the second round, for a two-round total of 0. We conclude that, after initially playing [PT1=100%,PT2=0%], the leader has 50% chance each of receiving total payoff of either 12 or 0, so the leader's expected total payoff is 6. Similarly, one can derive that the expected total utility of the leader strategy [PT1=0%,PT2=100%] in the first round of the game (refer to Tables (d) and (e)) is EU([PT1=0%,PT2=100%])=4.5.

Unfortunately, finding the optimal leader strategy in the first round of the game requires one to evaluate all the strategies [PT1=x,PT2=1-x], $0 \leq x \leq 1$, considering for each strategy the unique knowledge of the opponent payoff structure gained by observing the follower responses to the said strategies. As the derivation of the formulae for the the expected utilities of the leader actions for arbitrary repeated Stackelberg games is an open research problem, we propose to approach the problem using a customized version of the Monte-Carlo Tree Search method, as shown next.

3. MCTS APPROACH

3.1 MCTS Overview

Monte-Carlo Tree Search (MCTS) methods provide new tools for online planning in complex sequential decision problems that have generated considerable excitement in recent years, due to breakthrough results in challenging domains such as 19×19 Go [5] and General Game Playing [3]. The key innovation of MCTS is to incorporate node evaluations within traditional tree search techniques that are based on

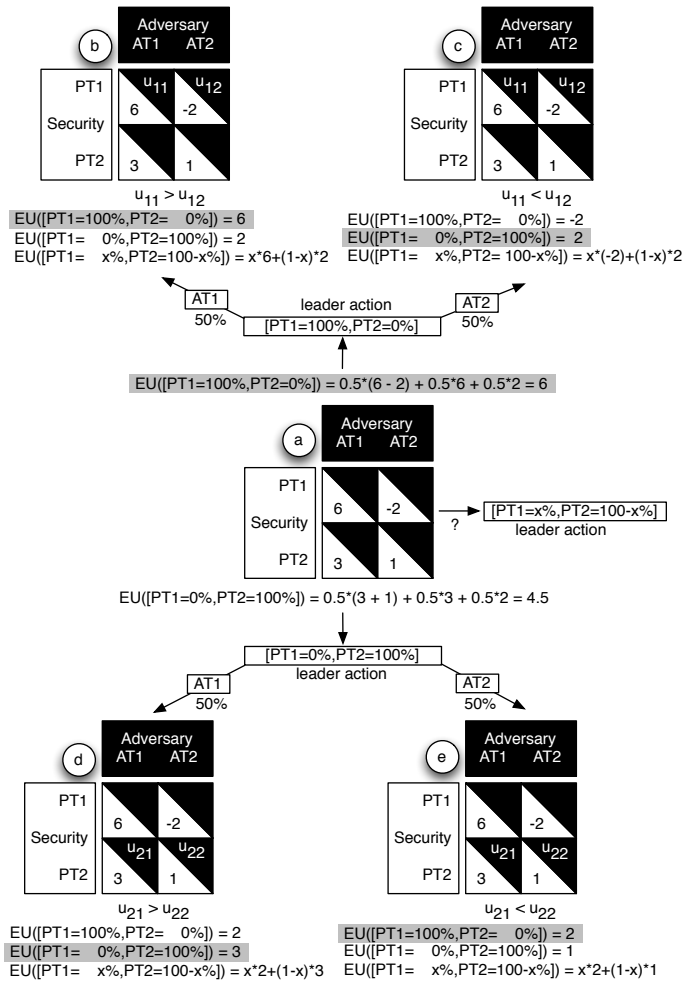

Figure 2: Two-round Stackelberg game, with uniformly distributed follower payoffs. First round of play shown in (a). If the leader plays 100% PT1, the follower may reply either AT1, leading to (b), or AT2, leading to (c). If the leader plays 100% PT2, follower may reply either AT1, leading to (d), or AT2, leading to (e).

stochastic simulations (i.e., "rollouts" or "playouts"), while also using bandit-sampling algorithms to focus the bulk of simulations on the most promising branches of the tree search. This combination appears to have overcome traditional exponential scaling limits to established planning techniques in a number of large-scale domains. MCTS is also an anytime algorithm and simple parallelization schemes have been found to scale effectively to hundreds of cores [16].

Standard implementations of MCTS maintain and incrementally grow a collection of nodes, usually organized in a tree structure, representing possible states that could be encountered in the given domain. The nodes maintain counts n_{sa} of the number of simulated trials in which action a was selected in state s, as well as mean reward statistics $\bar{r}_{sa}$ obtained in those trials. A simulation trial begins at the root node, representing the current state, and steps of the trial descend the tree using a tree-search policy that is based on sampling algorithms for multi-armed bandits that embody a tradeoff between exploiting actions with high mean reward, and exploring actions with low sample counts. When the trial reaches the frontier of the tree, it may continue performing simulation steps by switching to a "playout policy," which commonly select actions using a combination of heuristics. When the trial terminates, sample counts and mean reward values are updated in all tree nodes that participated in the trial. At the end of all simulations, the reward-maximizing top-level action from the root of the tree is selected and performed in the real domain.

Our implementation of MCTS makes use of the UCT algorithm [8], which employs a tree-search policy based on a variant of the UCB1 bandit-sampling algorithm [2]. The policy computes an upper confidence bound B_{sa} for each possible action a in a given state s according to: $B_{sa} = \bar{r}_{sa} + c\sqrt{\ln N_s / n_{sa}}$, where $N_s = \sum_{a'} n_{sa'}$ is the total number of trials of all actions in the given state, and c is a tunable constant controlling the tradeoff between exploration and exploitation. With an appropriate choice of the value of c, UCT is guaranteed to converge to selecting the best top-level action with probability 1.

3.2 MCTS in Repeated Stackelberg Games

We now present our MCTS-based method for planning leader actions in repeated Stackelberg games with unknown opponents. A key feature of our method builds upon the assumption that the leader has a prior probability distribution over possible follower types (equivalently, over follower utility functions). We leverage this by performing MCTS trials in which each trial simulates the behavior of the follower using an independent draw from this distribution. As different follower types transition down different branches of the MCTS tree, this provides a simple and elegant means of implicitly approximating the posterior distribution for any given history in the tree, where the most accurate posteriors are focused on the most critical paths for optimal planning. This may enable faster approximately optimal planning than established methods which require fully specified transition models for all possible histories as input to the method.

A high-level depiction of the method is given in Figure 3. The method performs a total of T simulated trials, each with a randomly drawn follower, where a trial consists of H rounds of play. In each round, the leader chooses a mixed strategy $\sigma \in \Sigma$ to be performed, that is, to play each pure strategy $a_l \in A_l$ with probability $\sigma(a_l)$. To obtain a finite enumeration of leader mixed strategies, similarly to [11], we discretize the $\sigma(a_l)$ values into integer multiples of a discretization interval $\epsilon = 1/K$, and represent the leader mixed strategy components as $\sigma(a_l) = k_l \cdot \epsilon$ where $\{k_l\}$ is a set of non-negative integers s.t. $\sum k_l = K$. In the example in Figure 3 the number of leader *pure* strategies is $|A_l| = 2$ and $K = 2$ and the leader can choose to perform only one of the following three *mixed* strategies: $LA1 = [0.0, 1.0]$; $LA2 = [0.5, 0.5]$ or $LA3 = [1.0, 0.0]$ as shown in Figure 3. Upon observing the leader mixed strategy, the follower then plays a greedy pure-strategy response, that is, it selects from among its pure strategies (FR1, FR2, FR3 in Figure 3) the strategy achieving highest expected payoff given the observed leader mixed strategy. Although such discretization method can in theory lead to suboptimal solutions [11], the underlying discretization error is rarely seen in practice [12], especially for big values of K. An argument can also be made that in real world Stackelberg games the leader can imple-

ment its mixed strategy (and the follower can observe it) with only a limited precision [14].

Leader strategies in each round of each trial are selected by MCTS using either the UCB1 tree-search policy for the initial rounds within the tree, or a playout policy for the remaining rounds taking place outside the tree. Our playout policy uses uniform random selection of leader mixed strategies for each remaining round of the playout. We grow the MCTS tree incrementally with each trial, starting from just the root node at the first trial. Whenever a new leader mixed strategy is tried from a given node, the set of all possible transition nodes (i.e. leader mixed strategy followed by all possible follower responses) are added to the tree.

The basic idea of the abbreviated proof of convergence of our algorithm is to map a repeated Bayesian Stackelberg game to an equivalent Markov Decision Process. The MDP states in such a mapping are the finite horizon histories of pairs of (discretized) leader mixed strategies and their corresponding follower responses. The MDP actions represent possible leader mixed strategies in a current state. The transition probability from state $s = (h)$ to state $s' = (h|\sigma a_f)$ given action $\sigma \in \Sigma$ equals the probability of a follower response a_f to σ which can be uniquely determined from $P(\theta)$ and the observed history h. Finally, the corresponding MDP reward is given in Equation 2 wherein $B(\theta, \sigma)$ from Equation 1 is known to be a_f. Following Lemma 1 in [17], the expected payoff of a repeated Bayesian Stackelberg game policy equals the expected payoff of the equivalent MDP policy and therefore the optimal repeated Bayesian Stackelberg game policy and the optimal MDP policy are equivalent. Since our method samples according to the UCT formula, which is guaranteed to converge to the optimal MDP policy [8], therefore it also converges to the optimal repeated Bayesian Stackelberg game policy.

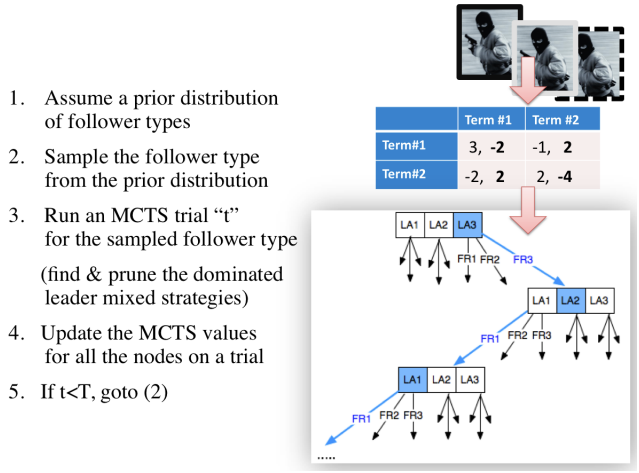

Figure 3: MCTS algorithm overview

4. PRUNING OF THE DOMINATED LEADER STRATEGIES

As it is shown in this section, in some cases, the leader's exploration of the complete reward structure of the follower is unnecessary. In essence, in any round of the game, the leader can identify the leader strategies—that have not yet been employed by the leader—whose immediate expected value for the leader is guaranteed not to exceed the expected value of leader strategies employed by the leader in the earlier rounds of the game. If the leader then just wants to maximize the expected payoff of its next action, these not-yet-employed strategies can safely be disregarded.

To formalize the concept of pruning of dominated leader strategies assume that the leader is playing a repeated Stackelberg game with a follower of type $\theta \in \Theta$. Furthermore, denote by $E^{(n)} \subset \Sigma$ a set of leader mixed strategies that have been employed by the leader in rounds $1, 2, ..., n$ of the game. Notice, that the leader who aims to maximize its payoff in the $n+1st$ round of the game should consider to employ an unused strategy $\sigma \in \Sigma - E^{(n)}$ only if:

$$\overline{U}(\theta, \sigma) > \max_{\sigma' \in E^{(n)}} U(\theta, \sigma') \tag{4}$$

Where $\overline{U}(\theta, \sigma)$ is the upper bound on the expected utility of the leader playing σ, established from the leader observations $B(\theta, \sigma')$; $\sigma' \in E^{(n)}$ as follows:

$$\overline{U}(\theta, \sigma) = \max_{a_f \in A_f(\sigma)} U(\sigma, a_f). \tag{5}$$

Where $A_f(\sigma) \subset A_f$ is defined here as a set of follower actions a_f that can still (given $B(\theta, \sigma')$; $\sigma' \in E^{(n)}$) constitute the follower best response to σ while $U(\sigma, a_f)$ is the expected utility of the leader mixed strategy σ if the follower responds to it by executing action a_f. That is:

$$U(\sigma, a_f) = \sum_{a_l \in A_l} \sigma(a_l) u_l(a_l, a_f) \tag{6}$$

Thus, in order to determine whether a not-yet-employed strategy σ should be executed, one has to determine the elements of a best response set $A_f(\sigma)$ given $B(\theta, \sigma')$ for all $\sigma' \in E^{(n)}$. We now show how that can be accomplished.

4.1 Best Response Sets

To find the actions that can still constitute the best response of the follower of type θ to a given leader strategy σ, we first define the concept of *Best Response Sets* and *Best Response Anti-Sets* and then prove an important property of best response sets.

DEFINITION 1. *For each action $a_f \in A_f$ of the follower, a best response set Σ_{a_f} is a set of all the leader strategies $\sigma \in \Sigma$ for which it holds that $B(\theta, \sigma) = a_f$.*

DEFINITION 2. *For each action $a_f \in A_f$ of the follower, a best response anti-set $\overline{\Sigma}_{a_f}$ is a set of all the leader strategies $\sigma \in \Sigma$ for which it holds that $B(\theta, \sigma) \neq a_f$.*

PROPOSITION 1. *Each best response set Σ_{a_f} is convex and $\{\Sigma_{a_f}\}_{a_f \in A_f}$ is a finite covering of Σ.*

PROOF. By contradiction: If Σ_{a_f} is not convex then there must exist $\sigma', \sigma'' \in \Sigma_{a_f}$ such that $B(\theta, \sigma') = B(\theta, \sigma'') = a_f$ and $\sigma = \lambda\sigma' + (1-\lambda)\sigma''$; $\lambda > 0$ such that $\sigma \notin \Sigma_{a_f}$. From Equation (1) it then holds that:

$$\begin{aligned}
\sum_{a_l \in A_l} \sigma'(a_l) u_f(a_l, a_f) &> \sum_{a_l \in A_l} \sigma'(a_l) u_f(a_l, \overline{a}_f) \\
\sum_{a_l \in A_l} \sigma''(a_l) u_f(a_l, a_f) &> \sum_{a_l \in A_l} \sigma''(a_l) u_f(a_l, \overline{a}_f) \\
\sum_{a_l \in A_l} \sigma(a_l) u_f(a_l, \overline{a}_f) &> \sum_{a_l \in A_l} \sigma(a_l) u_f(a_l, a_f).
\end{aligned}$$

Where $\overline{a}_f \in A_f$ is not a_f. After adding these inequalities and substituting $\sigma := \lambda\sigma' + (1-\lambda)\sigma''$ we obtain:

$$\sum_{a_l \in A_l} \sigma(a_l) u_f(a_l, \overline{a}_f) \quad < \quad \sum_{a_l \in A_l} \sigma(a_l) u_f(a_l, a_f)$$

Which contradicts the earlier inequality. Now, since for each $\sigma \in \Sigma$ there exists some $a_f \in A_f$ such that $B(\theta, \sigma) = a_f$, we have that $\{\Sigma_{a_f}\}_{a_f \in A_f}$ covers the entire set Σ and is therefore a partitioning of Σ. □

We first illustrate how to find the follower best responses on an example and then provide a method that achieves it in a general case. Specifically, we now illustrate that (after a few rounds of the games) there may indeed exist $\sigma \in \Sigma$ such that $A_f(\sigma) \neq A_f$. Consider an example in Figure 4 where the game has already been played for two rounds. Let $A_l = \{a_{l_1}, a_{l_2}\}$, $A_f = \{a_{f_1}, a_{f_2}, a_{f_3}\}$ and $E^{(2)} = \{\sigma', \sigma''\}$ where $\sigma'(a_{l_1}) = 0.25; \sigma'(a_{l_2}) = 0.75$ and $\sigma''(a_{l_1}) = 0.75; \sigma''(a_{l_2}) = 0.25$. Furthermore, assume $U(a_{l_1}, a_{f_1}) = 0$; $U(a_{l_2}, a_{f_1}) = 1$; $U(a_{l_1}, a_{f_2}) = 1$; $U(a_{l_2}, a_{f_2}) = 0$ and $U(a_{l_1}, a_{f_3}) = U(a_{l_2}, a_{f_3}) = 0$. The follower best responses observed so far are $B(\theta, \sigma') = a_{f_1}$ (solid black circle) and $B(\theta, \sigma'') = a_{f_2}$ (black circle with dashed perimeter).

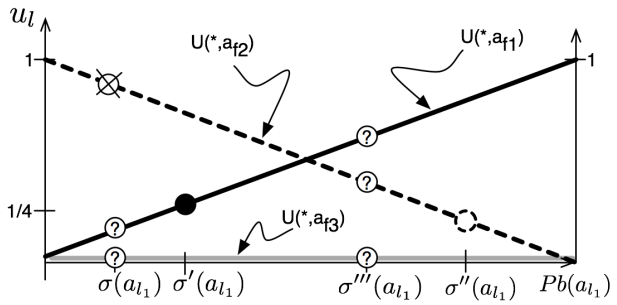

Figure 4: Best response actions

Notice, how in this context it is not profitable for the leader to employ a mixed strategy σ such that $\sigma(a_{l_1}) \in [0, \sigma'(a_{l_1})) \cup (\sigma''(a_{l_1}), 1]$. In particular, for σ such that $\sigma(a_{l_1}) \in [0, \sigma'(a_{l_1}))$ (refer to Figure 4 point σ) it holds that $B(\theta, \sigma) \neq a_{f_2}$ because otherwise (from Proposition (1)) the convex set $\Sigma_{a_{f_2}}$ would contain the elements σ and σ''—and hence also contain the element σ'—which is not true as $B(\theta, \sigma') = a_{f_1} \neq a_{f_2}$. Consequently, we have $A_f(\sigma) = \{a_{f_1}, a_{f_3}\}$ (notice the points with question marks above σ in Figure 4) which implies that $\overline{U}(\theta, \sigma) = \max\{U(\sigma, a_{f_1}), U(\sigma, a_{f_3})\} < \max\{0.25, 0\} = 0.25 = \max\{U(\sigma', a_{f_1}), U(\sigma'', a_{f_2})\}$. Hence, while employing strategy σ would allow the leader to learn $B(\theta, \sigma)$ (i.e., to disambiguate in Figure 4 the question marks in points above σ), this knowledge would not translate into the leader higher payoffs: The immediate expected reward for the leader for employing strategies σ', σ'' is always greater than the expected reward for employing σ such that $\sigma(a_{l_1}) \in [0, \sigma'(a_{l_1})) \cup (\sigma''(a_{l_1}), 1]$.

The example in Figure 4 also illustrates how the leader has to balance the benefits of exploration versus exploitation in the current round of the game. Specifically, the leader has a choice to either play one of the strategies σ', σ'' it had employed in the past (σ' if $U(\sigma', a_{f_1}) > U(\sigma'', a_{f_2})$ or σ'' otherwise) or play some strategy σ''' such that $\sigma'''(a_{l_1}) \in (\sigma'(a_{l_1}), \sigma''(a_{l_1})) = [0,1] \setminus [0, \sigma'(a_{l_1})) \setminus (\sigma''(a_{l_1}), 1]$ that it had not yet employed—and hence does not know what the follower best response $B(\theta, \sigma''')$ for this strategy is. Notice, that in this case, $A_f(\sigma''') = \{a_{f_1}, a_{f_2}, a_{f_3}\}$ (illustrated in Figure 4 by three points with question marks above σ'''). Now, if $B(\theta, \sigma''') = a_{f_3}$ were true, it would mean that $U(\sigma''', a_{f_3}) < \max\{U(\sigma', a_{f_1}), U(\sigma'', a_{f_2})\}$. In such case, the leader would explore the follower payoff preference (by learning $B(\theta, \sigma''')$) at a cost of loosing the potential immediate payoff of $U(\sigma''', a_{f_3}) - \max\{U(\sigma', a_{f_1}), U(\sigma'', a_{f_2})\}$.

Finally, the example also shows that although the immediate expected utility for executing a not-yet-employed strategy is smaller than the immediate expected utility for executing a strategy employed in the past, in some cases it might be profitable *not* to prune such not-yet-employed strategy. For example, if the game in Figure 4 is going to be played for at least two more rounds, the leader might still have an incentive to play σ, because if it turns out that $B(\theta, \sigma) = a_{f_3}$ then (from Proposition 1) $B(\theta, \sigma''') \neq a_{f_3}$ and consequently $\overline{U}(\theta, \sigma''') > \max\{U(\sigma', a_{f_1}), U(\sigma'', a_{f_2})\}$. In essence, if the execution of a dominated strategy can provide information about the follower preferences that will become critical in subsequent rounds of the game, one pruning heuristic might be to not prune such dominated strategy.

4.1.1 The Pruning Algorithm

When an MCTS trial starts (at the root node), the leader does not know how the follower is going to respond to *any* of its mixed strategies $\sigma \in \Sigma$, for it does not know the type $\theta \in \Theta$ of the follower that it is playing with. That is, the leader knows nothing about the sets Σ_{a_f} and anti-sets $\overline{\Sigma}_{a_f}$; $a_f \in A_f$. As the game enters subsequent rounds though, the leader collects the information about the follower responses to the leader strategies, assembles this information to infer more about Σ_{a_f} and $\overline{\Sigma}_{a_f}$; $a_f \in A_f$ and then prunes the provably dominated leader strategies that do not provide critical information to be used in later rounds of the game.

The pruning algorithm runs orthogonally to MCTS and can be applied to any MCTS node whose parent has already been serviced by the pruning algorithm. Consider one such MCTS node corresponding to a situation where the rounds $1, 2, ..., k-1$ of the game have already been played and let $\Sigma^{(k-1)} \subset \Sigma$ denote the set of leader strategies that have not yet been pruned (not to be confused with the set $E^{(k-1)}$ of leader strategies employed in rounds $1, 2, ..., k-1$ of the game). We have $\Sigma^{(0)} = \Sigma$ at the MCTS root node. Also, let $\Sigma_{a_f}^{(k-1)} \subset \Sigma_{a_f}$ and $\overline{\Sigma}_{a_f}^{(k-1)} \subset \overline{\Sigma}_{a_f}$ be the partially uncovered follower best response sets and anti-sets, inferred by the leader from its observations of the follower responses in rounds $1, 2, ..., k-1$ of the game. (Unless $|A_f| = 1$, we have $\Sigma_{a_f}^{(0)} = \emptyset$, $\overline{\Sigma}_{a_f}^{(0)} = \emptyset$; $a_f \in A_f$ at the MCTS root node.) When the leader then plays $\sigma \in \Sigma^{(k-1)}$ in the k-*th* round of the game and observes the follower best response $b \in A_f$, it constructs the sets $\Sigma^{(k)}$, $\Sigma_{a_f}^{(k)}$, $\overline{\Sigma}_{a_f}^{(k)}$; $a_f \in A_f$ as described in Algorithm 1.

Algorithm 1 starts by cloning the non-pruned action set (line 1) and best response sets (lines 2 and 3). Then, in line 4, $\Sigma_b^{(k)}$ becomes the minimal convex hull that encompasses itself and the leader strategy σ (computed e.g. using a linear program). At this point (lines 5 and 6), the algorithm constructs the best response anti-sets, for each $b' \in A_f$. In particular: $\sigma' \notin \Sigma_{b'}^{(k)}$ is added to the anti-set $\overline{\Sigma}_{b'}^{(k)}$ if there exists a vector (σ', σ'') where $\sigma'' \in \Sigma_{b'}^{(k)}$ that intersects some

Algorithm 1

Input: σ, b, $\Sigma^{(k-1)}$, $\Sigma_{a_f}^{(k-1)}$; $a_f \in A_f$

Output: $\Sigma^{(k)}$, $\Sigma_{a_f}^{(k)}$, $a_f \in A_f$

1: $\Sigma^{(k)} \leftarrow \Sigma^{(k-1)}$
2: **for all** $b' \in A_f$ **do**
3: $\quad \Sigma_{b'}^{(k)} \leftarrow \Sigma_{b'}^{(k-1)}$
4: $\Sigma_b^{(k)} \leftarrow$ ConvexHull$(\Sigma_b^{(k)}, \sigma)$
5: **for all** $b' \in A_f$ **do**
6: $\quad \overline{\Sigma}_{b'}^{(k)} \leftarrow \{\sigma' \in \Sigma \setminus \Sigma_{b'}^{(k)}$ s.t. $(\lambda\sigma' + (1-\lambda)\sigma'') \in \Sigma_{a_f}^{(k)}$ for some $\lambda > 0$; $\sigma'' \in \Sigma_{b'}^{(k)}$ and $a_f \in A_f$; $a_f \neq b'\}$
7: $\sigma^* \leftarrow \arg\max[\sigma' \in \Sigma_b^{(k)}]$ $\{U(\sigma', b)\}$
8: $\Sigma^{(k)} \leftarrow \Sigma^{(k)} \setminus (\Sigma_b^{(k)} \setminus \{\sigma^*\})$
9: **for all** $\sigma \in \Sigma^{(k)} \setminus \cup_{a_f \in A_f} \Sigma_{a_f}^{(k)}$ **and all** $b \in A_f$ **do**
10: $\quad$ **if** $\sigma \in \cap_{a_f \in A_f \setminus \{b\}} \overline{\Sigma}_{f_a}^{(k)}$ **then**
11: $\quad\quad$ **goto** 4

set $\Sigma_{a_f}^{(k)}$; $a_f \neq b$ (else, $\Sigma_{b'}^{(k)} \cup \{\sigma'\}$ would not be convex, thus violating Proposition 1). Next (lines 7 and 8), the algorithm prunes from $\Sigma^{(k)}$ all the strategies that are strictly dominated by σ^*, for which the leader already knowns the best response $b \in A_f$ of the follower. (Notice that no further information about the follower preferences can be gained by pruning these actions.) Finally, the algorithm loops (line 9) over all the non-pruned leader strategies σ for which the best response of the follower is still unknown; In particular (line 10) if $b \in A_f$ is the only remaining plausible follower response to σ, it automatically becomes the best follower response to σ and the algorithm goes back to line 4 where it considers the response b to the leader strategy σ as if it was actually observed. The pruning algorithm terminates its servicing of an MCTS node once no further actions can be pruned from $\Sigma^{(k)}$. One can then identify the leader strategies to be pruned from Equations (4, 5, 6).

5. EXPERIMENTS

5.1 Basic Checks

We first performed a series of experiments aimed at checking the validity of MCTS generated policy. One of these experiments (refer to Figure 5) is in the airport security domain of Figure 1. We set the discretization interval of leader mixed strategies to 0.25, resulting in a total of 5 leader actions. We considered a 4 stage game and ran $1,000,000$ MCTS trials assuming (a) a follower whose payoffs are sampled from a uniform prior distribution and (b) a follower whose payoffs are sampled from some distribution that reflects conflicting preferences of the leader and follower agents. As can be seen in Figure 5, the resulting MCTS policies appear to conform to our intuitions. In particular, in the uniform prior distribution case, notice how the leader chooses to play [PT1=50%,PT2=50%] in the third round of the game, to effectively learn (and later take advantage of) the follower best response to [PT1=50%,PT2=50%], having learned in the earlier rounds of the game the follower responses to the leader pure strategies. Also, in the zero-sum prior distribution case, notice that when the follower is identified to prefer to AT1, the leader never chooses to play a pure strategy PT1 as this would result in Terminal 2 being left unguarded.

An illustration of typical rate-of-convergence behavior in

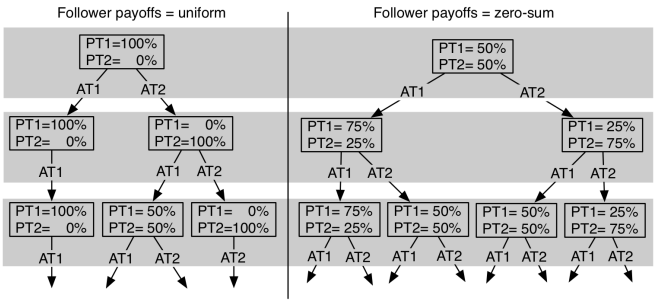

Figure 5: Policies generated by MCTS

our experiments is plotted in Figure 6. These experiments used followers with uniform random utility functions over 4 pure strategies, horizon $H = 6$, and leader discretization $\epsilon = 0.25$. The number of leader pure strategies is varied over $\{2, 3, 4\}$. We generally find clear convergence to the optimal policy and value estimate, at least in all of our small-scale studies. Note that the convergence time may not have a simple dependence on number of leader strategies, as it may also depend on specific details of the leader's utility function.

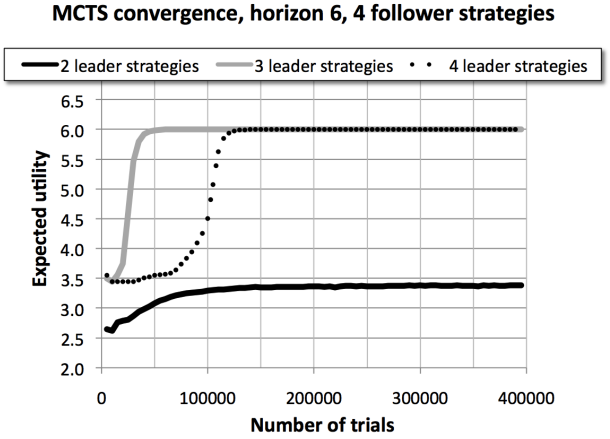

Figure 6: Value of best top-level action node vs. number of trials, illustrating typical MCTS convergence behavior.

5.2 Scaling studies without pruning

We present two scaling studies that examine how far we can push a "vanilla" version of MCTS with no built-in domain knowledge, and no capability of inference across nodes, so that the only way to estimate the value of a tree node is by explicit sampling. The first study focuses on scaling of convergence time with horizon H. We would expect MCTS convergence time to scale exponentially with H, possibly with a large base if the branching factor is large. However, certain aspects of the Bayesian Stackelberg domain could result in relatively mild scaling. First, it seems plausible that the optimal policy would consist of "exploratory" actions for the first few rounds, in order to determine the follower preferences, followed by pure exploitation actions that maximize immediate payoff in all remaining rounds. This means that it could be relatively easy for MCTS to find the best action at deep levels of the tree, as it would only need to find the action with best immediate payoff. This is easy to determine

in our domain, since the payoffs are deterministic given the leader and follower actions, and the follower type is likely to be uniquely determined deep in the tree.

Results of scaling of mean convergence time with H, from $H = 2$ to $H = 16$, are shown in Table 1. We define "convergence time" as the number of trials needed to reach 99% of asymptotic optimal value. We use uniform random follower utility functions, and fix the following experiment parameters: leader strategies = 3, follower strategies = 4, and discretization = 0.25.

Note that there are 15 possible leader mixed-strategy combinations, so the branching factor per round is nominally 60, including the possible follower responses. However, we can see much more favorable scaling in Table 1 than would be implied by the nominal branching factor.

Horizon	Trials to Converge	CPU (sec)
2	15,300	13.40
4	26,100	23.53
6	42,200	38.48
8	77,500	72.60
10	132,000	127.92
12	221,000	227.79
14	340,000	515.22
16	512,000	1124.75

Table 1: Number of MCTS trials to converge and CPU runtime as a function of horizon H.

Next, we examine scaling of convergence time in a fixed experiment configuration (uniform random followers, 3 leader and 4 follower strategies, and $H = 6$) where we vary the discretization interval ϵ. Here we expect that "vanilla" MCTS could perform quite poorly, as the branching factor increases exponentially in $1/\epsilon$, and MCTS has to explicitly sample each option to acquire any information regarding its value. Results shown in Table 2 indicate that the convergence time in fact blows up rapidly once ϵ becomes small, and we were unable to obtain convergence within one million trials for $\epsilon = 1/16$. The observed poor scaling with discretization provides ample motivation for using our pruning algorithm of Section 4, whose results are reported below.

Discretization	Trials to Converge	CPU (sec)
0.5	21,100	18.22
0.25	42,200	38.48
0.125	177,000	193.56
0.0625	—	—

Table 2: Mean convergence time (in thousands of trials and CPU runtime) as a function of discretization ϵ. The 0.0625 run failed to converge within one million trials.

5.3 Results using leader strategy pruning

In our final set of experiments we switched on the pruning of the dominated leader strategies to see how it improves the performance of our algorithm. We chose the number of leader and follower pure strategies to be $M = 2$ and $N = 2$ respectively and fixed the number of game rounds to $H = 10$. Across the three experiments in Figure 7 we progressively increased the discretization accuracy of the leader mixed strategies, by *decreasing* the length of the discretization interval from 25% to 12% and finally to 6%. For each setting we then ran a sufficient number of MCTS trials to obtain convergence, and plotted on the x-axis the current trial number and on the y-axis the expected utility of the currently best leader strategy.

All three experiments in Figure 7 show that using our pruning technique results in a clear reduction in the number of trials needed to obtain convergence. As can be seen in the first graph ($\epsilon = 1/4$), with pruning switched on, the algorithm converged to within 99% of its asymptotically optimal value in less than $100,000$ trials as compared to more than $200,000$ trials needed to accomplish the same task with pruning switched off. The impact of pruning is even more pronounced in the second graph ($\epsilon = 1/8$): Here, with pruning switched on, the algorithm managed to converge to the asymptotically optimal solution after approximately $670,000$ trials. In contrast, convergence with pruning switched off required more than 1.2 million trials, i.e., pruning reduced the required number of trials by over $500,000$. Finally, in the third graph in Figure 7, the increased accuracy of the discretization of the leader mixed strategies ($\epsilon = 1/16$) resulted in a game tree with 2^{50} nodes. The algorithm with pruning converged in ~ 4.2 million trials, representing a savings of 1M trials compared to the 5.2 million needed for convergence without pruning.

Of course, the number of trials needed for convergence is not the only relevant performance metric in these experiments, since the pruning technique entails greater computational overhead per trial. If the CPU cost of the pruning calculations were to exceed the savings in number of required trials, the pruning technique would not yield a net win in terms of wall-clock run time. Fortunately, our implementation of pruning limits the CPU cost of pruning to at most 50% of total simulation CPU time: we run pruning and MCTS search on two independent threads, with the pruning thread taking up to 50% of the cycles of a single CPU, and the search thread taking the remaining cycles. To obtain a fair comparison, the corresponding experiments without pruning also utilized two threads, each performing independent MCTS trials. As seen below in Table 3, the savings in number of trials more than compensates for the greater CPU overhead of the pruning technique. We observe a progressive increase in wall-clock time savings via pruning, from 25 seconds (157-132) at $\epsilon = 1/4$, to 94 seconds (637-540) at $\epsilon = 1/8$, to 277 seconds (2903-2626) at $\epsilon = 1/16$.

Discretization	Pruning Time	No-Pruning Time
0.25	132	157
0.125	540	637
0.0625	2626	2903

Table 3: Mean wall-clock time to converge, in seconds, as a function of discretization ϵ, in pruning vs. no-pruning experiments of Figure 7.

6. CONCLUSIONS

We presented the first-ever study of repeated Bayesian Stackelberg games where the objective is to maximize the leader's cumulative expected payoff over the rounds of the game. The optimal policies in such games must make intel-

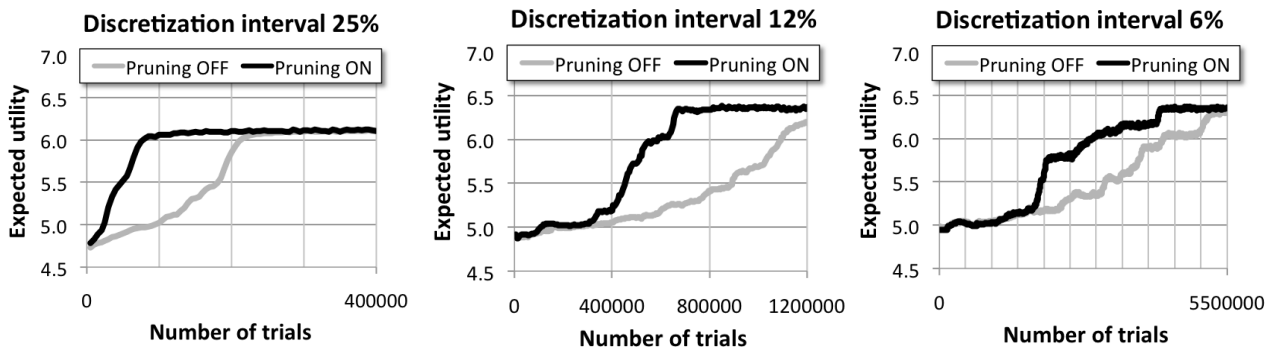

Figure 7: Impact of pruning of dominated leader strategies on the efficiency of MCTS

ligent tradeoffs between actions that reveal information regarding the unknown follower preferences, and actions that aim for high immediate payoff. We proposed an innovative approach to solve for such optimal policies, based on Monte Carlo Tree Search combined with an original method for pruning dominated leader strategies.

Our results show ample promise that the approach will be able to tackle numerous problems in real-world security domains. The points of particular promise that we see are: (1) MCTS trials based on sampling from the follower distribution appears to offer a highly efficient means of approximating follower posteriors over all possible history paths, insofar as they contribute to the optimal policy. (2) MCTS scaling is very manageable with respect to the number of rounds to be played. (3) Our pruning method addresses a major limitation of simple MCTS which scales poorly in the case of fine-grained discretization of leader mixed strategies.

Our findings draw support from the recent work by Silver and Veness [17] which obtained excellent scaling results in applying MCTS to solving large scale POMDPs. This suggests to us that the advantages of MCTS that we found in Bayesian Stackelberg Games may extend to more general classes of imperfect information games.

One of our future research studies will focus on generalizing the model to repeated Bayesian Stackelberg games where the follower also behaves strategically. In this more general case there might be scenarios where MCTS convergence may be much more difficult than what we found in the non-strategic case. This has been observed in recent work by Ramanujan et al. [15] which identified *soft traps*. We believe that such traps do not occur in our current formulation as the leader always gains information about the follower preferences. However, how to avoid such traps when facing a strategic follower player is a worthy topic of investigation.

7. ACKNOWLEDGMENTS

This work was supported in part under the DARPA GALE project, contract No. HR0011-08-C-0110.

8. REFERENCES

[1] T. Alpcan and T. Basar. A game theoretic approach to decision and analysis in network intrusion detection. In *42nd IEEE Conf. on Decision and Control*, 2003.

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

[3] H. Finnsson and Y. Björnsson. Simulation-based approach to general game playing. In *Proc. of AAAI-2008*, pages 259–264, 2008.

[4] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.

[5] S. Gelly et al. Modification of UCT with patterns in Monte-Carlo Go. Technical Report 6062, INRIA, 2006.

[6] M. Jain et al. Security games with arbitrary schedules. In *AAAI*, 2010.

[7] C. Kiekintveld, J. Marecki, and M. Tambe. Approximation methods for infinite bayesian stackelberg games: Modeling distributional payoff uncertainty. In *AAMAS*, 2011.

[8] L. Kocsis and C. Szepesvari. Bandit based Monte-Carlo planning. In *15th European Conf. on Machine Learning*, pages 282–293, 2006.

[9] J. Letchford, V. Conitzer, and K. Munagala. Learning and approximating the optimal strategy to commit to. In *Symp. on the Algorithmic Decision Theory*, 2009.

[10] K. Nguyen and T. Basar. Security games with incomplete information. In *IEEE Intl. Conf. on Communications*, 2009.

[11] P. Paruchuri et al. An efficient heuristic approach for security against multiple adversaries. In *AAMAS* , 2007.

[12] P. Paruchuri et al. Playing games with security: An efficient exact algorithm for Bayesian Stackelberg games. In *AAMAS*, 2008.

[13] J. Pita et al. Deployed ARMOR protection: The application of a game-theoretic model for security at the the LAX airport. In *AAMAS Industry Track*, 2008.

[14] J. Pita et al. Effective solutions for real-world stackelberg games: When agents must deal with human uncertainties. In *AAMAS*, 2009.

[15] R. Ramanujan, A. sabharwal, and B. Selman. Understanding sampling style adversarial search methods. In *Proceedings of UAI-2010*, 2010.

[16] R. Segal. On the scalability of parallel UCT. In H. van den Herik et al., editors, *Computers and Games*, volume 6515 of *LNCS*, pages 36–47. Springer Berlin / Heidelberg, 2011.

[17] D. Silver and J. Veness. Monte-Carlo Planning in Large POMDPs. In *Advances in Neural Information Processing Systems 22*, 2010.