Online Learning and Planning in Resource Conservation Games

Yundi Qian, William B. Haskell, Albert Xin Jiang, Milind Tambe
University of Southern California, Los Angeles, CA, 90089
{yundi.qian, william.b.haskell, jiangx, tambe}@usc.edu

ABSTRACT

Protecting our environment and natural resources is a major global challenge. "Protectors" (law enforcement agencies) try to protect these natural resources, while "extractors" (criminals) seek to exploit them. In many domains, such as illegal fishing, the extractors know more about the distribution and richness of the resources than the protectors, making it extremely difficult for the protectors to optimally allocate their assets for patrol and interdiction. Fortunately, extractors carry out frequent illegal extractions, so protectors can learn the richness of resources by observing the extractor's behavior. This paper presents an approach for allocating protector assets based on learning from extractors. We make the following four specific contributions: (i) we model resource conservation as a repeated game and transform this repeated game into a POMDP, which cannot be solved by the latest general POMDP solvers due to its exponential state space; (ii) in response, we propose GMOP, a dedicated algorithm that combines Gibbs sampling with Monte Carlo tree search for online planning in this POMDP; (iii) for a specific class of our game, we speed up the GMOP algorithm without sacrificing solution quality, as well as provide a heuristic that trades off solution quality for lower computational cost; (iv) we explore the continuous utility scenario where the POMDP becomes a continuous-state POMDP, and provide a solution in special cases.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

Keywords

Resource conservation; Repeated games; POMDP; Online planning

1. INTRODUCTION

Faced with the challenge of sustaining natural resources, many nations have established law enforcement agencies tasked with interdicting illegal extractions of natural resources. However, patrol assets are limited so that the allocation of these limited assets becomes a key challenge. This scarcity is also observed in security games for protecting critical infrastructure against terrorism [13]. However, protecting natural resources is very different. In security games, the attacker conducts extensive surveillance on the defender and executes a one-shot attack, while in resource conservation domains, the extractor also observes the protector's strategy but carries out frequent illegal extractions. We propose the term *resource conservation games* to refer to this new domain. In our paper, we refer to law enforcement as the "protector", environmental criminals as the "extractor", and locations with resources as "sites".

In resource conservation domains, the protector often does not know the distribution of resources while the extractor may have more information about it. This fact leads to the fundamental challenge addressed in out paper: how to provide an optimal asset deployment strategy for the protector, given her lack of knowledge about the distribution of resources? Similar planning problems exist in security games where the defender is uncertain about the attacker's utilities. Those work can be divided into two categories: (i) model uncertainty in terms of different attacker types and solve a Bayesian Stackelberg game [9, 15]; (ii) optimize the worst-case payoff without any assumptions about a prior distribution [4]. In resource conservation games, the extractor's frequent illegal extractions provide the protector with the opportunity to learn about the distribution of resources by observing the extractor's behavior. Our paper focuses on constructing an online policy for the protector to maximize her utility given observations of the extractor.

In this paper[1], we model this situation as a repeated game. We then recast this repeated game as a partially observable Markov decision process (POMDP). However, our POMDP formulation has an exponential number of states, making current POMDP solvers like ZMDP [12] and APPL [6] infeasible in terms of computational cost. Silver and Veness [11] have proposed the POMCP algorithm which offers good solution quality and is scalable to large POMDPs. It uses particle filters to maintain an approximation of the belief state, and then uses Monte Carlo Tree Search (MCTS) for online planning. However, the particle filter is only an approximation of the belief state. By appealing to the special properties of our POMDP, we propose the GMOP algorithm (**G**ibbs sampling based **M**CTS **O**nline **P**lanning) which draws samples directly from the exact belief state using Gibbs sampling and then runs MCTS for online planning. Our algorithm provides higher solution quality than the POMCP algorithm. Additionally, for a specific subclass of our game with an extractor who plays a best response against the protector's empirical distribution, and a uniform penalty of being seized across all sites, we provide an advanced sampling technique to speed up the GMOP algorithm along with a heuristic that trades off solution quality for lower computational cost. At last, we explore the continuous utility scenario where our original POMDP formulation becomes a continuous-state POMDP, which is generally difficult to solve. However, the special properties in the specific subclass of game mentioned above make possible the extension of our GMOP algorithm to continuous utility scenarios.

2. RELATED WORK

[1]This paper extends our AAMAS main track paper [10]. We extend our GMOP algorithm to the continuous utility scenario, and add more discussions about our experimental results.

There is a significant body of literature on single and multiagent learning. It can be divided into two categories: model-free approaches and model-based approaches. Q-learning [14] is a popular model-free approach that computes an optimal MDP policy without any prior knowledge about the reward and transition functions through trial and error. Model-free approaches make no assumptions about the opponent's strategy, but they learn slowly. Model-based approaches start with some model of the opponent's strategy, compute and play the best response, observe the opponent's play and then update the model of the opponent's strategy. The earliest such model-based approach is *fictitious play* [2], where the opponent is assumed to play a stationary mixed strategy and the first player assumes the opponent's mixed strategy is given by his empirical distribution. Conitzer and Sandholm propose the AWESOME algorithm [3] that: (i) converges to a Nash equilibrium when all other players use AWESOME; (ii) converges to a best response with probability 1 if all other players play stationary strategies. Unlike most existing model-based approaches which learn the opponent's strategy under the assumption of known utility functions, our model-based approach learns the opponent's utility function.

There is previous work on repeated games with incomplete information [1]. In these works, it is assumed that both players are strategic and play a Nash equilbrium. However, the extractors in the real world are not game theorists who play sophisticated strategies. Indeed, the current consensus from behavioral sciences suggest that human behavior is far from rational.

There has been previous work on learning attacker payoffs in repeated security games [7, 8]. Letchford et al. [7] develop an algorithm to uncover the attacker type in as few rounds as possible, while our paper focuses on maximizing the protector's utility. Marecki et al. [8] use MCTS to maximize the defender's utility in the first few rounds. However, their algorithm is unable to offer guidance in later rounds because it does not allow for belief updating, which is a major component of our paper. Additionally, Letchford et al. [7] and Marecki et al. [8] both assume that the defender plays a mixed strategy and the attacker plays a pure strategy that maximizes his expected utility given the defender's mixed strategy. However, illegal extractions happen frequently in resource conservation games, so the assumption that the extractor carries out surveillance over a long time to know the exact mixed strategy of the protector does not hold. Furthermore, we relax the assumption that the attacker is perfectly rational to handle more general behavior models such as quantal response.

3. MODEL

3.1 Motivating Domain

Our work is motivated by the domain of resource conservation, for example, illegal fishing, illegal oil extraction, water theft, crop theft, and illegal diamond mining, etc. In each case, illegal extractions happen frequently and the resources are spread over a large area that is impossible for the protector to cover in its entirety.

In our model, we make the assumption that the protector and the extractor fully observe their opponent's actions. The protector is usually a powerful government agency that has access to satellite imaging, multiple patrol assets, and the reports of local residents. The extractor learns about law enforcement tactics by exchanging information internally, covert observation, and by buying information from other sources.

Our flagship example is the real-world problem faced by the U.S. Coast Guard (USCG) in the Gulf of Mexico of illegal fishing by fishermen from across-the-border. In this domain, the protector (USCG) performs daily aircraft patrol surveillance[2]; satellites are also used to monitor illegal fishing[3]. Furthermore, illegal fishermen have well-organized support from across-the-border; USCG provided evidence that fishermen have surveillance on USCG boats.

3.2 Formal Model

We now formalize the preceding story into a two-player repeated game between a protector and an extractor. In our model, the amount of resources at each site will be fixed and the extractor will have full knowledge of this distribution. The protector will have to learn this distribution by observing the extractor's behavior.

We operate over the finite time horizon $t \in \mathbb{T} \triangleq \{1, \ldots, T\}$. There are n sites indexed by $\mathbb{N} \triangleq \{1, 2, \ldots, n\}$ that represent the locations of the natural resource in question: the extractor wants to steal resources from these sites and the protector wants to interdict the extractor. We represent the value of the sites to the extractor in terms of their utilities. Each site has a utility $u(i)$ that is only known to the extractor. The utility space is discretized into m levels, $u(i) \in \mathbb{M} \triangleq \{1, 2, \ldots, m\}$ since human beings cannot distinguish between tiny differences in utilities in the real world. For n sites and m utility levels, there are m^n possible sets of utilities across all sites. The distribution of resources is then captured by the vector of utilities at each site, and the set of possible resource distributions is:

$$\mathbb{U} \triangleq \{(u(1), u(2), \ldots, u(n)) : u(i) \in \mathbb{M}, \forall i \in \mathbb{N}\} = \times_{i \in \mathbb{N}} \mathbb{M}. \quad (1)$$

At the beginning of the game, the protector have some prior knowledge about the resource levels $u(i)$ at each site $i \in \mathbb{N}$. This prior knowledge is represented as a probability density function $p(u(i))$ over $\mathbb{M}$. If the protector does not know anything about $u(i)$, then we adopt a uniform prior for $u(i)$ over $\mathbb{M}$.

At each time $t \in \mathbb{T}$, the protector chooses a site $a_t \in \mathbb{N}$ to protect and the extractor simultaneously chooses a site $o_t \in \mathbb{N}$ from which to steal. If $a_t = o_t$, the protector catches the extractor and the extractor is penalized by the amount $P(o_t) < 0$; if $a_t \neq o_t$, the extractor successfully steals resources from site o_t and gets a payoff of $u(o_t)$. Additionally, both the protector and the extractor fully observe the moves of their opponent. Note that the penalty $P(i), i \in \mathbb{N}$ is known to both the protector and the extractor. We adopt a zero-sum game, so the protector is trying to minimize the extractor's payoffs. In most resource conservation domains, the extractor pays the same penalty P if he is seized independent of the site he visits. We allow for varying penalties across sites for greater generality.

In this work, we assume a fictitious Quantal Response playing (FQR) extractor. Specifically, a fictitious extractor assumes the protector's empirical distribution will be his mixed strategy in the next round. The extractor behaves in the following way: in every round, he (i) computes the empirical coverage probability c_i for every site i based on the history of the protector's actions; (ii) computes the expected utility $EU(i) = c(i)P(i) + (1 - c(i))u(i)$ for every site; (iii) attempts to steal from the site i with the probability proportional to $e^{\lambda EU(i)}$ where $\lambda \geq 0$ is the parameter representing the rationality of the player (higher λ represents a more rational player). The reader will see that our methodology can be applied to other human behavior models, for example, the extractor can be fully rational and visit the site with the highest expected utility (a special case of QR corresponding to taking the limit $\lambda \to \infty$); the extractor can visit sites with expected utilities that exceed a cer-

[2]http://www.uscg.mil/d8/sectCorpusChristi/

[3]http://wwf.panda.org/?206301/WWF-new-approach-to-fight-illegal-unreported-and-unregulated-fishing

tain threshold; the extractor can have only a limited memory of the protector's actions, etc.

3.3 Protector's POMDP Formulation

There are two technical questions to resolve in our model: First, at every round t, based on her current belief about u, how should the protector choose sites to protect in the next round? Second, after each round, how should the protector use the observation of the latest round to update her beliefs about u? We are studying decision making and belief updating in a partially observable environment where the payoffs u are unobservable and the extractor's actions are observable, which is the exact setup for a POMDP. We now setup our two-player game as a POMDP $\{\mathbb{S}, \mathbb{A}, \mathbb{O}, T, \Omega, R\}$.

state space The state space of our POMDP is $\mathbb{S} = \mathbb{U} \times \mathbb{Z}^n$, which is the cross product of the utility space and the count space. $\mathbb{U}$ is the utility space as defined in Equation 1. $\mathbb{Z}^n$ is the set of possible counts of the protector's visits to each site, where $C_t \in \mathbb{Z}^n$ is an integer-valued vector where $C_t(i), i \in \mathbb{N}$ is the number of times that the protector has protected site i at the beginning of round $t \in \mathbb{T}$. A particular state $s \in \mathbb{S}$ is written as $s = (u, C)$, where u is the vector of utility levels for each site and C is the current state count. The initial beliefs are expressed by a distribution over $s = (u, 0)$, induced by the prior distribution on u. We define $c_t(i) \triangleq \frac{C_t(i)}{t-1}$ to be the frequency with which the protector visits site i at the beginning of round $t \in \mathbb{T}$. We set $c_1 \triangleq 0$ by convention.

action space The action space $\mathbb{A}$ is $\mathbb{N}$, representing the site the protector chooses to protect.

observation space The observation space $\mathbb{O}$ is $\mathbb{N}$, representing the site the extractor chooses to attempt to steal from.

conditional transition probability Let $e_a \in \mathbb{R}^n$ denote the unit vector with a 1 in slot $a \in \mathbb{N}$ and zeros elsewhere. The conditional transition probability T governing the evolution of the state is

$$T\left(s' = \left(u',\, C'\right) \mid s = (u, C),\, a\right) = \begin{cases} 1, & u = u',\, C' = C + e_a, \\ 0, & \text{otherwise.} \end{cases}$$

Specifically, the evolution of the state is deterministic. The underlying utilities do not change, and the count for the site visited by the protector increases by one while all others stay the same.

conditional observation probability We define $EU(u, C) \in \mathbb{R}^n$ to be the vector of empirical expected utilities for the extractor for all sites when the actual utility is u and the count is C,

$$[EU(u, C)](i) = c(i)P(i) + (1 - c(i))u(i), \forall i \in \mathbb{N},$$

when $t \geq 1$. We set $[EU\,(u,\, 0)]\,(i) = u(i)$ by convention. Hence, our observation probabilities Ω are explicitly

$$\Omega(o|s' = (u, C), a) = \frac{e^{\lambda[EU(u, C - e_a)](o)}}{\sum_{i \in \mathbb{N}} e^{\lambda[EU(u, C - e_a)](i)}},$$

the probability of observing the extractor takes action o when the protector takes action a and arrives at state s'. Note that both a and o are the actions the protector/extractor take at the same round.

reward function The reward function R is

$$R\left(s = (u, C), s' = (u, C + e_a), a, o\right) = \begin{cases} -P(o), & a = o, \\ -u\,(o)\,, & a \neq o. \end{cases}$$

We conclude this section by briefly explaining how the above POMDP can be modified to allow for other extractor behavior models. We only require the extractor's decisions to be a function of the utilities u, penalty P, and the count C. We can then calculate the probability that the extractor visits site $i, i \in \mathbb{N}$ in every possible state $s = (u, C)$. Thus, we can accordingly modify the conditional observation probabilities Ω to treat this extractor behavior, while the rest of the POMDP is unchanged. If the extractor's decision making depends on the sequence of the protector's actions more than the count C, for example, if the extractor has limited memory or weighs recent protector activity more heavily, then the state space also needs to be modified to become $\mathbb{S} = \mathbb{U} \times \{\mathbb{N}^i : i \in \{0, 1, 2, \ldots, T\}\}$. $\mathbb{U}$ is still the utility space and $\{\mathbb{N}^i : i \in \{0, 1, 2, \ldots, T\}\}$ is now the entire history of the protector's actions. Note when the extractor has a limited memory k we only need to keep track of the last $k \leq T$ time steps.

4. GMOP ALGORITHM

In Section 3, we modeled our repeated game as a POMDP. However, the size of the utility space $\mathbb{U}$ is m^n, and the size of the count space is $\mathcal{O}(\frac{T^n}{n!})$. The computational cost of the latest POMDP solvers such as ZMDP and APPL soon become unaffordable for us as the problem size grows. For a small instance like $n = 4$, $m = 5$ and 5 rounds, there are 78750 states in the POMDP. Both the ZMDP and APPL solvers run out of memory when attempting to solve this POMDP. This challenge is non-trivial because our models in reality are much larger than this toy example.

Silver and Veness [11] have proposed the POMCP algorithm, which provides high quality solutions for large POMDPs. The POMCP algorithm uses a particle filter to approximate the belief state. Then, it uses Monte Carlo tree search (MCTS) for online planning where (i) state samples are drawn from the particle filter and (ii) the action with the highest expected utility based on Monte Carlo simulations is chosen. However, the particle filter is only an approximation of the true belief state and is likely to move further away from the actual belief state as the game goes on, especially when most particles get depleted and new particles need to be added. Adding new particles will either (i) make the particle filter a worse approximation of the exact belief state, if the added particles do not follow the distribution of the belief state or (ii) be as difficult as drawing samples directly from the belief state, if the added particles do follow the distribution of the belief state. However, if we could efficiently draw samples directly from the exact belief state, then there would be no need to use a particle filter.

Our POMDP has specific structure—the count state in $\mathbb{S}$ is known and the utility state does not change, making it possible to draw samples directly from the exact belief state. We propose the GMOP algorithm that draws samples directly from the exact belief state using Gibbs sampling, and then runs MCTS. The samples drawn directly from the belief state better represent the true belief state compared to samples drawn from a particle filter. We thus conjecture that our GMOP algorithm will yield higher solution quality than the POMCP algorithm for our problem, and this intuition is confirmed in our experiments.

4.1 GMOP Algorithm Framework

The GMOP algorithm is outlined in Algorithm 1. At a high level, in round t the protector draws samples of state s from its belief state $B_t(s)$ using Gibbs sampling and then it runs MCTS using those samples. Finally, it executes the action with the highest expected utility. MCTS starts with a tree that only contains a root node. Since the count state C_t is already known, the protector only needs to sample the utility state u from B_t. The sampled state s is comprised of the sampled utility u and the count C_t.

It has been shown that the UCT algorithm converges to the optimal value function in fully observable MDPs [5]. Based on this result, Silver and Veness have established the convergence of MCTS in POMDP online planning as long as the samples are drawn from the true belief state $B_t(s)$. It follows that the convergence of our

Algorithm 1 GMOP Algorithm Framework

```
function PLAY(C_t)
    Initialize Tree
    for i = 1 → numSamples do
        u ← GIBBSSAMPLING
        SIMULATE(s = (u, C_t))
    end for
    a_t ← best action in tree
end function
```

GMOP algorithm is guaranteed.

From Algorithm 1, we see that each iteration of our algorithm is composed of two parts: GIBBSSAMPLING which draws samples u directly from $B_t(u)$ using Gibbs sampling, and SIMULATE which does Monte Carlo simulation of the sampled states $s = (u, C_t)$. Our sampling technique will be discussed in detail in Section 4.2 while the details of MCTS in POMDP are available in [11].

4.2 Drawing Samples

4.2.1 Gibbs Sampling Overview

Gibbs sampling is a Markov chain Monte Carlo (MCMC) algorithm for sampling from multivariate probability distributions. Let $X = (x_1, x_2, \ldots, x_n)$ be a general random vector with n components and with finite support described by the multivariate probability density $p(X)$. Gibbs sampling only requires the conditional probabilities $p(x_i|x_{-i})$ to simulate X, where $x_{-i} = (x_j)_{j\neq i}$ denotes the subset of all components of X except component i. Gibbs sampling is useful when direct sampling from $p(X)$ is difficult.

Suppose we want to obtain k samples of $X = (x_1, x_2, \ldots, x_n)$. Algorithm 2 shows how Gibbs sampling works in general to produce these samples using only the conditional probabilities $p(x_i|x_{-i})$. It constructs a Markov chain whose steady-state distribution is given by $p(X)$, so that the samples we draw also follow the distribution $p(X)$. The states of this Markov chain are the possible realizations of $X = (x_1, x_2, \ldots, x_n)$, and a specific state X_i is denoted as $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ (there are finitely many such states by our assumption). The transition probabilities of this Markov chain, $Pr(X_j|X_i)$, follow from the conditional probabilities $p(x_i|x_{-i})$. Specifically, $Pr(X_j|X_i) = p(x_l|x_{-l})$ when $x_{jv} = x_{iv}$ for all v not equal to l, and is equal to zero otherwise, i.e. the state transitions only change one component of the vector-valued sample at a time. This Markov chain is reversible (meaning $p(X_i)Pr(X_j|X_i) = P(X_j)Pr(X_i|X_j), \forall i, j$) so $p(X)$ is its steady-state distribution.

Algorithm 2 Gibbs Sampling

```
Initialization: X = {x_1, x_2, ..., x_n} satisfying p(X) > 0
for i = 1 → k do
    for j = 1 → n do
        x_j ~ p(x_j|x_{-j})
    end for
    X^i ← {x_1, x_2, ..., x_n}
end for
```

4.2.2 Applying Gibbs Sampling in GMOP

We let B_t be the probability distribution representing the protector's beliefs about the true utilities at the beginning of round $t \geq 1$; B_1 represents the protector's prior beliefs when the game starts. We adopt the notation $B_t(u)$ to denote the probability of the vector of utilities u with respect to the distribution B_t.

Let B be the prior belief distribution and B' be the posterior belief distribution. Our Bayesian belief update rule to obtain B' from B and the observation is explicitly

$$\begin{aligned} B'(s' = (u, C)) =& \eta\Omega(o|s', a)\sum_{s\in\mathbb{S}} T(s'|s, a)B(s) \\ =& \eta\Omega(o|s', a)B(s = (u, C - e_a)). \end{aligned}$$

If a_t and o_t represent the actions that the protector and the extractor choose to take at round t, we have

$$\begin{aligned} B_t(u) =& \eta B_{t-1}(u)\Omega(o_{t-1}|s = (u, C_t), a_{t-1}) \\ =& \eta' B_1(u)\Pi_{i=1}^{t-1}\Omega(o_i|s = (u, C_{i+1}), a_i). \end{aligned} \quad (2)$$

It follows that the posterior belief B_t is proportional to the prior belief B_1 multiplied by the observation probabilities over the entire history. Since there are m^n possible utilities, it is impossible to store and update B_t when m and n are large, and thus it is impossible to sample directly from B_t. Hence, we turn to Gibbs sampling. We only need the conditional probabilities $p(u_i|u_{-i}), \forall i$ in B_t

$$\begin{aligned} & p(u_i|u_{-i}) = \eta p(u_i, u_{-i}) = \eta B_t(u_i, u_{-i}) \\ =& \eta' B_1(u_i, u_{-i})\Pi_{j=1}^{t-1}\Omega(o_j|s = (u = (u_i, u_{-i}), C_{j+1}), a_j) \\ =& \eta'' B_1(u_i)\Pi_{j=1}^{t-1}\Omega(o_j|s = (u = (u_i, u_{-i}), C_{j+1}), a_j). \end{aligned} \quad (3)$$

This quantity is easy to compute where $B_1(u_i)$ is the prior probability that site i has utility u_i. Besides the conditional probability, we also need to find a valid u with $B_t(u) > 0$ to initialize Gibbs sampling. Finding such a u is easy in our FQR model because any u with $B_1(u) > 0$ satisfies $B_t(u) > 0$. In other behavior models, where finding a valid u is not so intuitive, one possibility is to check the sampled utilities at the latest round to pick a valid one.

5. FICTITIOUS BEST RESPONSE

In this section, we focus attention on a limiting case of the FQR model, a fictitious best response playing (FBR) extractor. An FBR extractor plays a best response against the empirical distribution of the protector, a similar assumption is found in [7, 8]. Additionally, we assume that all sites share the same penalty P, this assumption is satisfied in most resource conservation games. We will see that these two assumptions allow us to greatly speed up the GMOP algorithm. We also put forward a computationally inexpensive heuristic that offers high quality solutions.

When the extractor is FBR, our POMDP is roughly the same as in the FQR case except that the conditional observation probabilities Ω are now

$$\Omega(o|s' = (u, C), a) = \begin{cases} \frac{1}{|A(u, C-e_a)|}, & o \in A(u, C - e_a), \\ 0, & \text{otherwise}, \end{cases}$$

where $A(u, C)$ is the set of sites with maximal empirical expected utility when the actual utility is u and the count is C, i.e.

$$A(u, C) = \arg\max_{i\in\mathbb{N}} [EU(u, C)](i) \subset \mathbb{N},$$

The FBR extractor is actually a limiting case of the more general FQR model, we obtain this case by taking $\lambda \to \infty$. If we run the POMCP algorithm for an FBR extractor, the particles produced by the particle filter will be depleted very quickly and most utility states will take on probability 0 after only a few rounds. For example, if $n = 10$ and the protector observes that the extractor visits site 3 in the first round, then approximately 90% of possible utility states take on probability 0. Compared with FQR, more new particles must be added in the FBR case. Thus, the particle

filter is a worse approximation of the belief state, leading to worse performance of the POMCP algorithm.

5.1 Speeding Up GMOP

Gibbs sampling requires computation of the conditional probability $p(u_i|u_{-i})$ as described in Equation 3. However, t grows as the game evolves and the computational cost increases linearly with t. Under our assumptions of an FBR extractor and uniform penalties, we can use an advanced algorithm to compute $p(u_i|u_{-i})$ with its computational cost bounded by constant time. Proofs of Proposition 1 and 2 are available in Qian et al. [10] and are omitted here due to the lack of space.

Define

$$I_t(i,j) \triangleq \begin{cases} I_{t-1}(i,j), & i \neq o_{t-1}, \\ \max\{I_{t-1}(i,j), \frac{1-c_{t-1}(j)}{1-c_{t-1}(i)}\}, & i = o_{t-1}, \end{cases}$$

and $I_1(i,j) \triangleq 0, \forall i,j \in \mathbb{N}$. The quantities $I_t(i,j)$ can be computed recursively from $I_{t-1}(i,j)$ at very little computational cost.

PROPOSITION 1. *For a specific u, $\Pi_{i=1}^{t-1}\Omega(o_i|s=(u,C_{i+1}),a_i) > 0 \iff \frac{u(i)-P}{u(j)-P} \geq I_t(i,j), \forall i,j \in \mathbb{N}$.*

By checking if u satisfies $\frac{u(i)-P}{u(j)-P} \geq I_t(i,j), \forall i,j \in \mathbb{N}$, we can figure out if $\Pi_{j=1}^{t-1}\Omega(o_j|s=(u,C_{j+1}),a_j)$ equals 0. We now explain how to compute $\Pi_{j=1}^{t-1}\Omega(o_j|s=(u,C_{j+1}),a_j)$ if it does not equal 0. Define $V_t(i) \triangleq \{k : o_k = i, k \in \{1,2,\ldots,t-1\}\}, \forall i \in \mathbb{N}$ to be the set of rounds where the extractor attempts to steal from site i; define $V_t^{eq}(i,j) \triangleq \{k : j \in A(u,C_k), k \in V_t(i)\}, \forall i,j \in \mathbb{N}$ to be the set of rounds where the extractor attempts to steal from site i, but where site j gives the extractor the same expected utility. We define $V_t^{neq}(i,j) \triangleq \{k : j \notin A(u,C_k), k \in V_t(i)\}, \forall i,j \in \mathbb{N}$ to be the set of rounds where the extractor attempts to steal from site i and site j gives the extractor lower expected utility. Additionally, we define

$$\text{Tie}_t(i,j) \triangleq \{k : I_t(i,j) = \frac{1-c_k(j)}{1-c_k(i)}, k \in V_t(i)\}, \forall i,j \in \mathbb{N}$$

Like $I_t(i,j)$, $\text{Tie}_t(i,j)$ can be computed recursively at very little cost. By definition, $V_t^{eq}(i,j) \cap V_t^{neq}(i,j) = \phi$, $V_t^{eq}(i,j) \cup V_t^{neq}(i,j) = V_t(i)$ and $\text{Tie}_t(i,j) \subseteq V_t(i), \forall i,j \in \mathbb{N}$.

PROPOSITION 2. *If $\frac{u(i)-P}{u(j)-P} = I_t(i,j)$, $V_t^{eq}(i,j) = Tie_t(i,j)$, $V_t^{neq}(i,j) = V_t(i) - Tie_t(i,j)$; If $\frac{u(i)-P}{u(j)-P} > I_t(i,j)$, $V_t^{neq}(i,j) = V_t(i)$, $V_t^{eq}(i,j) = \phi$.*

Algorithm 3 shows how our advanced sampling technique resamples $u(k)$ from the conditional probabilities $p(u_i|u_{-i})$ by using the quantities I_t and Tie_t. The input u is the current set of sampled utilities; k is the index of u to be resampled according to $p(u_k|u_{-k})$; and I and 'Tie' are the latest I and 'Tie' that have been computed recursively. We set $\#A(j) = |A(u,C_j)|$ to denote the number of sites that have maximal expected utility for the extractor at round j, and we initialize these quantities to be 1 because $o_k \in A(u,C_k)$ by definition. Then, we check every pair of sites $i,j \in \mathbb{N}$: (i) if $\frac{u(i)-P}{u(j)-P} < I_t(i,j)$, then we set $B_t(u) = 0$ according to Proposition 1; (ii) if $\frac{u(i)-P}{u(j)-P} = I_t(i,j)$, then we set $V_t^{eq}(i,j) = \text{Tie}_t(i,j)$ according to Proposition 2, and we increase $\#A(k)$ by 1 for those $k \in \text{Tie}_t(i,j)$ because $j \in A(u,C_k), \forall k \in V_t^{eq}(i,j) = \text{Tie}_t(i,j)$; (iii) if $\frac{u(i)-P}{u(j)-P} > I_t(i,j)$, then $V_t^{eq}(i,j) = \phi$ according to Proposition 2, so we do nothing. After checking all pairs $i,j \in \mathbb{N}$, we determine: (i) whether $B_t(u) = 0$ and (ii) $\#A(k), \forall k \in \{1,2,\ldots,t-1\}$ if $B_t(u) > 0$. Based on these evaluations, the conditional probability $Prob = p(u_i|u_{-i})$ used to resample $u(k)$ is computed according to Equation 3. Finally, $Prob$ is normalized and then we sample the new $u(k)$.

Algorithm 3 Advanced Sampling Technique

```
1: function DRAWSAMPLE(u, k, I, Tie)
2:     Prob = B_1(u_k)
3:     for i = 1 → m do
4:         u(k) ← i
5:         #A(j) ← 1, ∀j = 1 → currRound − 1
6:         for p = 1 → n, q = 1 → n do
7:             if (u(p)−k)/(u(q)−k) < I(p,q) then
8:                 Prob(i) ← 0
9:                 break
10:            else if (u(p)−k)/(u(q)−k) = I(p,q) then
11:                #A(j) ← #A(j) + 1, ∀j ∈ Tie(p,q)
12:            end if
13:        end for
14:        if Prob(i) ≠ 0 then
15:            Prob(i) ← Prob(i) * Π_{j=1}^{currRound−1} 1/#A(j)
16:        end if
17:    end for
18:    Normalize Prob
19:    u(k) ∼ Prob
20:    return u
21: end function
```

5.2 Myopic Planning Heuristic

For our GMOP algorithm, larger sample sizes in MCTS leads to higher solution quality but at the expense of greater computational cost. Some domains require decisions to be made very quickly, so the protector may get poor performance with the GMOP algorithm due to an insufficient number of samples. With this motivation, we provide a myopic planning heuristic. This heuristic offers slightly lower solution quality compared with GMOP, but costs much less computing time.

The myopic planning heuristic works as follows: it (i) approximately computes the posterior marginal probabilities of all sites' utilities based on all previous observations; (ii) computes the expected $u(i)$ for each site using the posterior marginal probabilities; (iii) plans myopically—protects the site with the highest estimated expected utility for the extractor based on the expected $u(i)$ computed in step (ii) and the empirical visit counts C (ties are broken with even probabilities).

The key issue lies in step (i)—the computation of the posterior marginal probabilities of the utilities $u(i)$. This step can be viewed as inference in a Bayesian network. An example Bayesian network where $n = 4$ is shown in Figure 1. Here, $u(i), \forall i \in \mathbb{N}$ are treated as the unobserved random variables in the Bayesian network, and they have prior probabilities $B_1(u(i))$ for all $i \in \mathbb{N}$. We define $f(u(i),u(j)), \forall i,j \in \mathbb{N}$ to be observable binary random variables that depend on $u(i), u(j)$:

$$f(u(i),u(j)) \triangleq \begin{cases} 1, & \frac{u(i)-P}{u(j)-P} \geq I_t(i,j), \frac{u(j)-P}{u(i)-P} \geq I_t(j,i), \\ 0, & \text{otherwise}, \end{cases}$$

In the Bayesian Network, we have observations of $f(u(i),u(j)) = 1, \forall i,j \in \mathbb{N}$ according to Proposition 1, and our aim is to infer the posterior marginal probabilities for $u(i), \forall i \in \mathbb{N}$. The

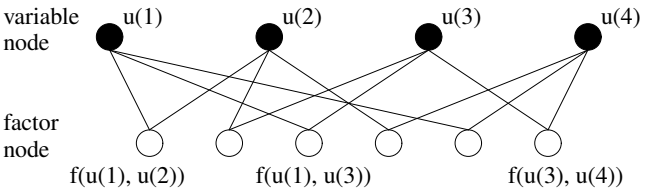


Figure 1: Bayesian Network when $n = 4$

widely known belief propagation algorithm is then used for inference, which yields approximate marginal probabilities.

Note that this heuristic does not take into consideration possible ties in the extractor's decision-making. In particular, recall that $\frac{u(i)-P}{u(j)-P} = I_t(i,j)$ and $\frac{u(i)-P}{u(j)-P} > I_t(i,j)$ correspond to two different cases, as we have shown in Proposition 2, and they are treated separately in Algorithm 3. Yet, the Bayesian network is unable to distinguish between these two cases and it treats them both as $\frac{u(i)-P}{u(j)-P} \geq I_t(i,j)$. Hence, the Bayesian network does not utilize all of the information that the protector has obtained and thus cannot offer an accurate description of the true belief state. Subsequently, the posterior probability we compute from the Bayesian network is inaccurate even if an exact inference algorithm is used. However, we note that in our experiments we get satisfactory solution quality even though both the Bayesian network formulation and the belief propagation algorithm are inexact.

6. CONTINUOUS UTILITY SCENARIO

In previous sections, we discussed the model with discretized utilities with the justification that humans can not distinguish between tiny differences so that it works as long as m is large enough. However, it is difficult to tell how "large" is large enough and larger m leads to more computational cost so that we can not increase m arbitrarily large. In this section, we try to extend our model to continuous utility scenario, i.e. $u(i) \in [0,1]$ so that it is expressive enough to describe human's perception of utilities.

When the utilities are discrete, we formulated a POMDP formulation in Section 3.3. When the utilities are continuous, this formulation remains the same except that the utility space $\mathbb{U}$ becomes

$$\mathbb{U} \triangleq \{(u(1), u(2), \ldots, u(n)) : u(i) \in [0,1], \forall i \in \mathbb{N}\}$$

which is in continuous space. Thus, our previous POMDP formulation becomes a continuous-state POMDP, which is harder to solve and lacks efficient solutions.

Our GMOP algorithm is composed of two steps: sampling from the utility space and running MCTS with those samples. In continuous utilities scenario, the latter step remains the same so that the key issue here is to sample from the continuous utility space, which involves the computation of the conditional probability as described in Equation 3. In general cases, this computation involves the multiplication of several functions ($\Pi_{j=1}^{t-1}\Omega(o_j|s = (u = (u_i, u_{-i}), C_{j+1}), a_j)$ with u_i as the variable), which is generally hard to compute unless those functions have special properties, e.g. when the extractor is a FBR extractor and all sites share the same penalty P.

PROPOSITION 3. *When the extractor is a FBR player and all sites share the same penalty P, $\Pi_{j=1}^{t-1}\Omega(o_j|s = (u = (u_i, u_{-i}), C_{j+1}), a_j)$ is a boxcar function and if the non-zero interval is $[a,b]$, $a = \max_{j\in\mathbb{N}, j\neq i}\{P + I_t(i,j)(u(j) - P)\}$, $b = \min_{j\in\mathbb{N}, j\neq i}\{P + \frac{u(j)-P}{I_t(j,i)}\}$.*

PROOF. When the extractor is a FBR player and all sites share the same penalty P, $\Omega(o_j|s = (u = (u_i, u_{-i}), C_{j+1}), a_j)$ are all boxcar functions, so that their multiplication is also a boxcar function.

Recall Proposition 1, $\Pi_{i=1}^{t-1}\Omega(o_i|s = (u, C_{i+1}), a_i) > 0 \iff \frac{u(i)-P}{u(j)-P} \geq I_t(i,j), \forall i,j \in \mathbb{N}$. In our situation, u_{-i} is given, and we are trying to find the smallest and largest u_i satisfying $\Pi_{i=1}^{t-1}\Omega(o_i|s = (u, C_{i+1}), a_i) > 0$, i.e. $\frac{u(i)-P}{u(j)-P} \geq I_t(i,j), \forall i,j \in \mathbb{N}$. So we have $u_i \geq P + (u_j - P)I_t(i,j), \forall j \in \mathbb{N}, j \neq i$ and $u_i \leq P + \frac{u_j - P}{I_t(j,i)}, \forall j \in \mathbb{N}, j \neq i$. □

With Proposition 3, we can compute $\Pi_{j=1}^{t-1}\Omega(o_j|s = (u = (u_i, u_{-i}), C_{j+1}), a_j)$ very efficiently, making it possible for us to draw samples from the continuous state space using Gibbs sampling and then to search for the best strategy with MCTS.

7. EXPERIMENTAL EVALUATION

We will evaluate the performance of our algorithms in this section through extensive numerical experiments. For most of our upcoming experiments we use the following settings: $n = 10$, $m = 10$ (discrete utility space), the penalty across all sites is $P = -50$, and the prior probability distribution $B_1(u_i), i \in \mathbb{N}$ is uniform. All results are averaged over 1000 simulation runs. For each simulation run, we randomly draw the true utilities $u(i), i \in \mathbb{N}$ and then simulate the actions the protector and the extractor would take over the rounds of the game. Solution quality is assessed in terms of the average reward that the protector gets in the first few rounds of the game. There are two parameters in MCTS for our GMOP algorithm: $numSamples$ is the number of samples and $maxHorizon$ is the depth of the tree, i.e. the number of horizons we look ahead in the POMDP.

GMOP vs ZMDP/APPL To begin, we compare our GMOP algorithm with the ZMDP solver [12] and the APPL solver [6], both are general POMDP solvers. For a small problem instance like $n = 4$, $m = 5$ and total rounds $maxHorizon = 5$, there are 78750 states in the POMDP. Both ZMDP and APPL solvers run out of memory in this small problem instance. Hence, we test the two solvers together on an even smaller instance with $n = 3$, $m = 5$, $P = -10$ and $maxHorizon = 5$, so that the resulting POMDP has only 7000 states. We also include the random policy where the protector randomly chooses one site to protect at each round as a baseline. Table 1 reports the average reward of these three algorithms for both the FQR ($\lambda = 0.5, 1$ and 1.5) and FBR extractor. In this table, the columns titled H_i for $i = 1, \ldots, 5$ represent the GMOP algorithm with $maxHorizon$ set to be i and $numSamples$ set to be 10000. We see that the GMOP algorithm with $maxHorizon$ varying from 1 to 5 and the APPL/ZMDP solvers are very close in terms of average reward, and they all outperform the random policy a lot.

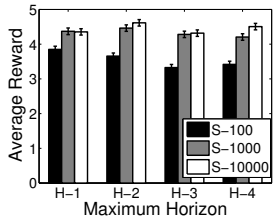
Table 1: ZMDP/APPL vs GMOP in Solution Quality

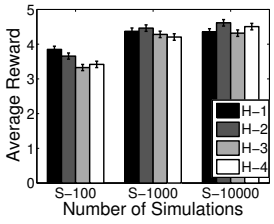
	Random	ZMDP	APPL	H_1	H_2	H_3	H_4	H_5
FQR(0.5)	1.13	3.85	3.85	3.90	3.89	3.95	3.90	3.91
FQR(1)	1.05	4.84	4.81	4.75	4.80	4.87	4.97	4.79
FQR(1.5)	1.03	5.35	5.39	5.35	5.36	5.42	5.36	5.34
FBR	1.09	6.32	6.31	6.25	6.24	6.27	6.32	6.36

Analysis of GMOP Now we investigate the effect of $numSamples$ and $maxHorizon$ on the performance of MCTS in our GMOP algorithm. Figure 2(a)/2(b) reports the results for the FQR model, Figure 3(a)/3(b) reports the results for the FBR model, and Figure 4(a)/4(b) reports the results for FBR model in continuous utility scenario. Figure 2(a)/3(a)/4(a) show that the performance of MCTS

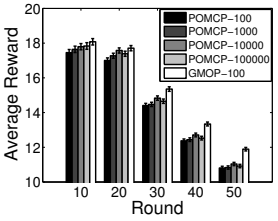
improves as we increase $numSamples$ while holding $maxHorizon$ fixed, demonstrating the convergence of MCTS. Figures 2(b)/3(b)/4(b) together show that: (i) if $numSamples$ is large enough to ensure convergence for both larger $maxHorizon$ and smaller $maxHorizon$ (10000 in discrete utility scenario), planning with more horizons ahead (larger $maxHorizon$) increases the reward the protector can get; (ii) if $numSamples$ is not large enough to ensure convergence for larger $maxHorizon$ (100 in discrete utility scenario and $100/1000/10000$ in continuous utility scenario), the reward the protector can get decreases as $maxHorizon$ increases because larger $maxHorizon$ indicates a deeper Monte Carlo tree so that more samples are needed to ensure convergence and it deteriorates the performance of MCTS if convergence is not reached. An interesting observation is that continuous utility scenario requires more samples because it involves a far larger utility space.

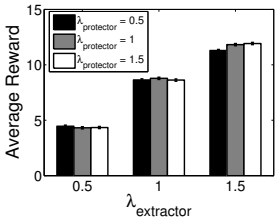
An interesting phenomenon observed in Figure 2(b)/3(b)/4(b) is that the performance difference between different $maxHorizon$ is tiny. However, this is not always the case. Here we provide an example where $maxHorizon$ makes a big difference in performance. Suppose that we have 3 sites: $u(1) = 5$; $u(2)$ is 10 with the probability 40% and 4 with the probability 60%; $u(3)$ has the same utility as $u(2)$. The penalties across all sites are all 0. We have 2 rounds in total. In round 1, If the protector chooses to protect site 1, the extractor gets $0.4 * 10 = 4$; if the protector chooses to protect site 2 or 3, the extractor gets $0.4 * 0.5 * 10 + 0.6 * 5 = 5$. Thus the protector will choose to protect site 1 if the $maxHorizon = 1$. In round 2, if the protector chooses to protect site 1 in round 1, the extractor steals from site 2 or 3 with equal probability, so he will get $0.5 * 0.6 * 4 + 0.5 * 0.4 * 10 = 3.2$, which gives the extractor the utility 7.2 in total; if the protector chooses to protect site 2 or 3 in round 1, the protector will learn exactly where the extractor is going to steal from, and the extractor gets 0, which gives the extractor the utility 5 in total. Thus the extractor will choose to protect site 2 or 3 in round 1 if $maxHorizon = 2$ and gets the expected utility -5 in these two rounds while getting the utility -7.2 if $maxHorizon = 1$.





(a) MCTS convergence (FQR: $\lambda = 0.5$, 50 rounds) (b) Different horizons (FQR: $\lambda = 0.5$, 50 rounds)

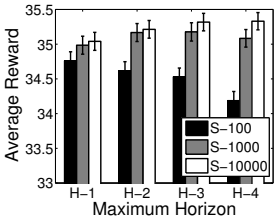


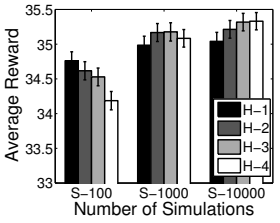


(c) GMOP vs POMCP in solution quality (FQR: $\lambda = 1.5$, $maxHorizon = 1$) (d) Robustness (FQR, 50 rounds, $numSamples = 1000$, $maxHorizon = 2$)

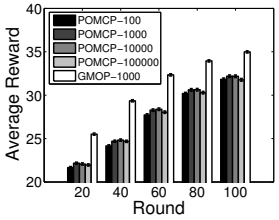
Figure 2: Fictitious Quantal Response

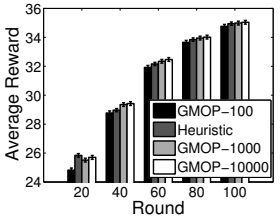
POMCP (Particle Filter) vs GMOP (Gibbs Sampling) In this paper we use Gibbs sampling to drive MCTS instead of the particle filter, as in the original POMCP algorithm [11]. In this way, the distribution of our samples is closer to the actual belief state. We now compare the performance of these two sampling techniques. The runtime of Gibbs sampling roughly increases linearly with $numSamples$; the runtime of the particle filter roughly increases linearly with the size of the particle filter (number of particles). For a fair comparison, we fix the particle filter size as well as $numSamples$ in Gibbs sampling.





(a) MCTS convergence (FBR, 100 rounds) (b) Different horizons (FBR, 100 rounds)





(c) GMOP vs POMCP in solution quality (FBR, $maxHorizon = 1$) (d) GMOP vs heuristic in solution quality (FBR)

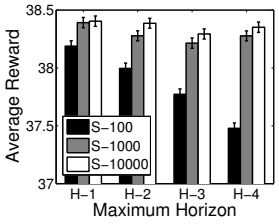
Figure 3: Fictitious Best Response

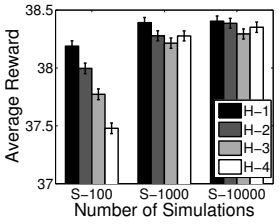
For the FQR model, we set the particle filter size to be 100000 and $numSamples$ in Gibbs sampling to be 100. The total runtimes are recorded in Table 2, where we see that the runtime of our GMOP algorithm is shorter than the runtime of POMCP as $numSamples$ varies from 100 to 100000. However, Figure 2(c) demonstrates that the performance of the GMOP algorithm with 100 samples exceeds the performance of the POMCP algorithm regardless of the value of $numSamples$. This performance gap between GMOP and POMCP grows with time because the particle filter gives an increasingly worse approximation of the belief state as time evolves.

Table 2: GMOP vs POMCP in Runtime(s)

GMOP-100	POMCP-100	POMCP-1000	POMCP-10000	POMCP-100000
31.71	75.86	72.92	75.89	92.26

Fictitious Quantal Response ($\lambda = 1.5$), $maxHorizon = 1$





(a) MCTS convergence (FBR, 100 rounds) (b) Different horizons (FBR, 100 rounds)

Figure 4: Continuous Utility Scenario

Table 3 and Figure 3(c) show the runtime and reward of GMOP with $numSamples = 1000$ vs POMCP with filter size 10000, for

the FBR extractor. For the FBR extractor, we see the same pattern but with an even larger gap in solution quality. In the FBR extractor model, the particles are depleted much more quickly than in the FQR model so that more new particles must be added, making the particle filter a worse approximation of the exact belief state.

Table 3: GMOP vs POMCP in Runtime(s)

GMOP-1000	POMCP-100	POMCP-1000	POMCP-10000	POMCP-100000
83.48	224.35	240.83	257.40	282.71

Fictitious Best Response, $maxHorizon = 1$

Robustness In some situations, the protector may not know the true value of λ which measures the extractor's rationality. In this experiment, we allow the extractor's true value of λ to take values in $0.5, 1, 1.5$, and we allow the protector to estimate λ to be any of $0.5, 1, 1.5$, for a total of 9 combinations of the true λ and its estimate. Figure 2(d) presents the results of this experiment. It turns out that the protector only does slightly worse when she incorrectly estimates the extractor's true λ.

Evaluation of the Advanced Sampling Technique in FBR Model In Section 5.1, we proposed an advanced way to compute conditional probabilities when using Gibbs sampling in the FBR model. This technique is less computationally expensive than the general method. Table 4 compares the runtimes of the general sampling technique with our advanced sampling technique. As the number of rounds increases from 20 to 100, the total runtime of the advanced sampling technique increases linearly, implying that the sampling cost at each round is approximately the same. On the other hand, the total runtime of the general sampling technique increases with the square of the number of rounds in the game, implying that the sampling cost is increasing linearly in each round.

Table 4: General vs Advanced Sampling in Runtime(s)

	20	40	60	80	100
General	51.77	209.31	469.80	835.15	1303.95
Advanced	43.83	62.04	77.24	92.77	108.67

Fictitious Best Response, $numSamples = 1000$, $maxHorizon = 1$

GMOP vs Myopic Planning Heuristic Our myopic heuristic trades solution quality for computational efficiency for a FBR extractor. Figure 3(d) compares the solution quality of the myopic planning heuristic versus GMOP, and Table 5 compares their total runtimes. For a fair comparison, we set $maxHorizon$ to be 1 in the GMOP algorithm. Figure 3(d) indicates that the heuristic gives better solutions than GMOP with $numSamples = 100$. However, the solution quality of the heuristic is worse than the one produced by GMOP when $numSamples$ equals 1000 or 10000. According to Table 5, the runtime of the myopic heuristic is much less than the runtime of GMOP.

Table 5: GMOP vs Heuristic in Runtime(s)

Heuristic	GMOP-100	GMOP-1000	GMOP-10000
0.49	8.38	83.48	689.87

Fictitious Best Response

8. CONCLUSION

This paper presents an online planning algorithm for the protector in resource conservation games. Our algorithm uses the information gained by observing the extractor's actions. We model this problem as a repeated game and then cast it as a POMDP. The latest POMDP solvers fail to scale up to our problem, so in response, we propose the GMOP algorithm which is based on MCTS and Gibbs sampling. Our experimental results show that our algorithm provides the same solution quality as the general POMDP solvers like ZMDP/APPL and outperforms the previous POMCP algorithm. Additionally, for the special case with FBR extractor and uniform penalties, we provide an advanced sampling technique to speed up the GMOP algorithm, a heuristic that trades off solution quality for lower computational cost and an extension of the GMOP algorithm to the continuous utility scenario.

9. ACKNOWLEDGEMENT

This research is supported by the United States Department of Homeland Security through the National Center for Risk and Economic Analysis of Terrorism Events (CREATE) under award number 2010-ST-061-RE0001 and MURI grant W911NF-11-1-0332.

10. REFERENCES

[1] R. J. Aumann and M. B. Maschler. *Repeated games with incomplete information*. The MIT press, 1995.

[2] G. W. Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 1951.

[3] V. Conitzer and T. Sandholm. Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning*, 2007.

[4] C. Kiekintveld, T. Islam, and V. Kreinovich. Security games with interval uncertainty. In *AAMAS*, 2013.

[5] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*.

[6] H. Kurniawati, D. Hsu, and W. S. Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008.

[7] J. Letchford, V. Conitzer, and K. Munagala. Learning and approximating the optimal strategy to commit to. In *Algorithmic Game Theory*. Springer, 2009.

[8] J. Marecki, G. Tesauro, and R. Segal. Playing repeated stackelberg games with unknown opponents. In *AAMAS*, 2012.

[9] P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S. Kraus. Playing games for security: an efficient exact algorithm for solving bayesian stackelberg games. In *AAMAS*, 2008.

[10] Y. Qian, W. B. Haskell, A. X. Jiang, and M. Tambe. Online planning for optimal protector strategies in resource conservation games. In *AAMAS*, 2014.

[11] D. Silver and J. Veness. Monte-carlo planning in large pomdps. In *NIPS*, 2010.

[12] T. Smith. *Probabilistic Planning for Robotic Exploration*. PhD thesis, Carnegie Mellon University, 2007.

[13] M. Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 2011.

[14] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 1992.

[15] Z. Yin and M. Tambe. A unified method for handling discrete and continuous uncertainty in bayesian stackelberg games. In *AAMAS*, 2012.