Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

# Scalable Initial State Interdiction for Factored MDPs

**Swetasudha Panda** and **Yevgeniy Vorobeychik**
Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN
{swetasudha.panda,yevgeniy.vorobeychik}@vanderbilt.edu

## Abstract

We propose a novel Stackelberg game model of MDP interdiction in which the defender modifies the initial state of the planner, who then responds by computing an optimal policy starting with that state. We first develop a novel approach for MDP interdiction in factored state space that allows the defender to modify the initial state. The resulting approach can be computationally expensive for large factored MDPs. To address this, we develop several interdiction algorithms that leverage variations of reinforcement learning using both linear and non-linear function approximation. Finally, we extend the interdiction framework to consider a Bayesian interdiction problem in which the interdictor is uncertain about some of the planner's initial state features. Extensive experiments demonstrate the effectiveness of our approaches.

## 1 Introduction

Stackelberg games have been widely used in recent years to model the strategic interaction between a defender and an attacker in security problems in a variety of settings [Jain *et al.*, 2008; Korzhyk *et al.*, 2011; Paruchuri *et al.*, 2008]. Examples include physical security (such as air marshall assignment and coast guard patrols), sustainability (such as security measures to prevent poaching), and computer network security. Commonly in such models the defender assigns security resources to targets, and the attacker chooses which target(s) to attack. However, in many settings the attacker performs a series of activities to accomplish their malicious goal. For example, in physical security these may involve reconnaissance, the choice of equipment to bring, and the path to take to targets which is taken. A general way to capture such multi-stage attacks is planning [Boddy *et al.*, 2005; Obes *et al.*, 2013; Ammann *et al.*, 2002; Ng *et al.*, 2010; Krautsevich *et al.*, 2012]. The defensive actions in such a scenario can then be viewed as *plan interdiction* [Letchford and Vorobeychik, 2013], whereby a defender deploys protective measures to compromise the ability of the attacker to successfully execute its plan.

Letchford and Vorobeychik [2013] first introduced a Stackelberg game framework for plan interdiction, where the defender removes a subset of attack actions and the adversary computes an optimal attack plan in the restricted action space. While interdiction of deterministic plans was quite scalable, the approach scaled poorly when modeling uncertainty with Markov Decision Processes (MDPs). Panda and Vorobeychik [2017] proposed an interdiction algorithm for MDPs with a factored representation of states. However, even this approach suffers from significant scalability limitations, particularly in capturing uncertainty about the attacker.

We propose a novel interdiction model in which the defender modifies the initial state of the attacker. This is quite general: for example, we can model prior interdiction approaches by adding action-specific preconditions as state variables. However, we show that this change enables significantly simpler and far more scalable interdiction techniques which rely on single-level integer linear programming, in contrast to difficult bi-level problems faced in prior art. We further improve scalability by using model-free reinforcement learning techniques with linear and non-linear action-value function approximators. In the former, we make use of a Fourier basis approximation for Boolean functions, develop methods for iteratively constructing a small subset of basis functions, and formulate an integer linear program for optimal interdiction. For the latter, we propose two iterative local search methods, the first optimizing one variable at a time, and the second using iterative linear approximations. Finally, we present a natural extension of our interdiction framework to Bayesian interdiction, in which the defender is uncertain about parameters of the attacker's planning problem.

We demonstrate the effectiveness and scalability of our proposed approaches compared to baseline alternatives on realistic examples from the international planning competition.

## 2 Preliminaries

**MDPs and Factored MDPs** A discounted infinite-horizon MDP is defined as a tuple $\mathcal{M} = (\mathbf{X}, A, R, P, \gamma)$ where $\mathbf{X}$ is a finite set of $|\mathbf{X}|$ states; $A$ is a finite set of actions; $R$ is a *reward function* $R : \mathbf{X} \times A \mapsto \mathbb{R}$, in which $R(\mathbf{x}, a)$ is the reward obtained by the agent in state $\mathbf{x}$ after taking action $a$; $P$ is a *Markovian transition model* where $P(\mathbf{x}'|\mathbf{x}, a)$ is the probability of moving from state $\mathbf{x}$ to $\mathbf{x}'$, after taking action $a$; and $\gamma \in [0, 1)$ is the discount factor which exponentially discounts future rewards. Such MDPs always admit an optimal *stationary deterministic policy*, which is a map-

ping $\pi : \mathbf{X} \mapsto A$, where $\pi(\mathbf{x})$ is the action the agent takes at state $\mathbf{x}$ [Puterman, 1994]. Each policy is associated with a *value function* $\mathcal{V}_\pi(\mathbf{x})$: the discounted cumulative value obtained by starting at state $\mathbf{x}$ and following policy $\pi$. Formally, $\mathcal{V}_\pi(\mathbf{x}) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R(\mathbf{X}^t, \pi(\mathbf{X}^t))|\mathbf{x}^{(0)} = \mathbf{x}]$, where $\mathbf{X}^{(t)}$ represents the state after $t$ steps.

The optimal policy $\pi^*$ can be computed as the greedy policy with respect to the optimal value function $\mathcal{V}^*$, $\pi^* = \arg\max_a [R(\mathbf{x}, a) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}'|\mathbf{x}, a)\mathcal{V}^*(\mathbf{x}')]$. The optimal action-value function $Q^*(\mathbf{x}, a)$ is the expected utility of taking action $a$ in state $\mathbf{x}$ and following the optimal policy thereafter. The discounted reward MDP is a natural model for security, since attackers prefer to achieve their goals (positive rewards) earlier, and incur costs (negative rewards) later. Additionally, the attack which takes too many steps has far more opportunities to fail in practice.

*Factored MDPs* exploit structure for compact representation. The set of states is described by a set of $m$ random state variables $\mathbf{X} = \{X_1, \ldots, X_m\}$. The transition model is represented as the product of local factors by using a DBN and the reward function as the sum of a set of localized reward functions. A factored (linear) value function ([Koller and Parr, 1999]) $\mathcal{V}$ is a *linear function* over a set of *basis functions* $\mathcal{B} = \{\phi_1, \ldots, \phi_k\}$, such that $\mathcal{V}(\mathbf{x}) = \sum_{j=1}^{k} w_j \phi_j(\mathbf{x})$ for some coefficients $\mathbf{w}$, where the scope of each $\phi_i$ is restricted to some subset of variables. The constant basis with empty scope is always included for feasibility. The approximate LP is given by Guestrin *et al.* [2003]:

$$\text{minimize} \sum_j \alpha_j w_j, \text{ subject to} \tag{1a}$$

$$\forall a, \max_{\mathbf{x}} \{R^a(\mathbf{x}) + \sum_j w_j [\gamma g_j^a(\mathbf{x}) - \phi_j(\mathbf{x})]\} \leq 0 \tag{1b}$$

where $R^a(\mathbf{x})$ is the factored reward, $\alpha_j = \sum_{\mathbf{x}} \alpha(\mathbf{x})\phi_j(\mathbf{x})$ for some *state relevance* parameters $\alpha(\mathbf{x}) > 0$ with $\sum_{\mathbf{x}} \alpha(\mathbf{x}) = 1$, and $g_j^a(\mathbf{x}) = \sum_{\mathbf{x}'} P(\mathbf{x}'|\mathbf{x}, a)\phi_j(\mathbf{x}')$ is the factored expected future value.

**Learning the Value Function with Function Approximation** In model-free reinforcement learning (RL) algorithms (e.g., Q-learning [Watkins and Dayan, 1992; Sutton and Barto, 1998]), the agent interacts with the environment in a sequence of actions with the goal of maximizing future rewards. The value function is directly approximated using the Bellman equation as an iterative update $Q'(\mathbf{x}^t, a) = \mathbb{E}[r + \gamma \max_{a'} Q(\mathbf{x}^{t+1}, a')|\mathbf{x}, a]$. To incorporate generalization, linear and non-linear function approximation are commonly used [Sutton and Barto, 1998]. Recently, popular RL with deep neural networks algorithms store the agent's data in memory and sample random batches for learning (*experience replay*) to deal with correlated data, non-stationary distributions, and convergence issues in learning with neural networks [Mnih *et al.*, 2013].

## 3 MDP State Interdiction

We model MDP interdiction as a Stackelberg (two-stage, one-shot) game with two players: defender and attacker. The defender is the Stackelberg leader, choosing an optimal interdiction policy. In our model, and unlike prior work, the defender transforms a given *initial state* of the MDP, $\mathbf{x}_0$ (which represents an initial or default configuration) into a new *start state*, $\mathbf{x}'_0$ of the attacker's MDP. Note that this model is quite general. For example, we can capture removing an action by adding to it a binary precondition such that the action is not applicable when this precondition is false; removing this action is then equivalent to negating this precondition in $\mathbf{x}'_0$

The attacker is the follower, and computes an MDP policy beginning with the start state $\mathbf{x}'_0$ chosen by the defender. In many settings of interest, such as cybersecurity, the state variable domains are finite and relatively small (indeed, prior work has modeled it using deterministic planning [Boddy *et al.*, 2005]), and we can therefore represent the associated planning problems as MDPs with binary variables. Consequently, we henceforth assume that the state space is comprised of a collection of binary variables.

Formally, the *MDP state interdiction problem (MDPSI)* is defined by an MDP $\mathcal{M} = \{\mathbf{X}, A, R(\mathbf{x}, a), \gamma\}$, where $\mathbf{X}, A, R(\mathbf{x}, a), \gamma$ are the state space, action space, reward function, and the discount factor of an infinite-horizon discounted MDP which the attacker is solving, as well as the defender's reward function $R_D(\mathbf{x}, a)$ and interdiction costs $\rho(\mathbf{x}'_0, \mathbf{x}_0) = \sum_i \rho_i |x'_{i0} - x_{i0}|$ of modifying an initial state $\mathbf{x}_0$ into $\mathbf{x}'_0$, where $\rho_i$ is the cost of modifying variable $i$. Note that this cost can also capture inability to modify specific state variables (the corresponding cost is then infinite).

We assume that the game is zero-sum (modulo interdiction costs), so that $R_D(\mathbf{x}, a) = -R(\mathbf{x}, a)$. Define $\mathcal{V}(\mathbf{x}_0, \pi)$ as the attacker's value function for a policy $\pi$ starting at state $\mathbf{x}_0$ in MDP $\mathcal{M}$. Let $\Pi$ be the set of deterministic stationary policies of the MDP. The defender then solves

$$\min_{\mathbf{x}'_0 \in \mathbf{X}} \mathcal{V}(\mathbf{x}'_0, \pi^*) + \rho(\mathbf{x}'_0, \mathbf{x}_0),$$

where $\pi^* \in \arg\max_{\pi \in \Pi} \mathcal{V}(\mathbf{x}_0, \pi)$ is the optimal policy of $\mathcal{M}$. The key bottleneck is the exponential state space in the attacker MDP. We propose reasonable approximations for the attacker's value function and scalable algorithms to compute it using a) factored MDP solution approaches and b) reinforcement learning. The key observation is that the optimal policy is *independent* of the interdiction decision, since the solution to an MDP is a function of an arbitrary state; to put it differently, the attacker's optimal policy is *already* a function of the defender's choice of the start state, by virtue of it solving an MDP. This is a key to significantly cleaner optimization problems for MDP interdiction below compared to prior art, as well as associated scalability gains.

## 4 Integer Linear Program for Approximately Optimal Interdiction

As a first approach, we use a linear approximation of the attacker's value function, and leverage a linear programming approach for approximate planning in factored MDPs. In addition, we make use of a Fourier representation of functions over a Boolean hypercube as the basis. In particular, any function $f : \{0, 1\}^m \to \mathbb{R}$ can be uniquely represented as $f(\mathbf{x}) = \sum_{j:S_j \subseteq \{1,\ldots,m\}} \hat{f}(S_j)\phi_j(\mathbf{x})$, where $\phi_j$ is a parity

function over the subset $S_j$ of variables [O'Donnell, 2008]:

$$\phi_j(\mathbf{x}) = \prod_{i \in S_j} (-1)^{x_i} = \begin{cases} +1, & \text{if } \sum_{i \in S_j} x_i \bmod 2 = 0. \\ -1, & \text{if } \sum_{i \in S_j} x_i \bmod 2 = 1. \end{cases} \quad (2)$$

We define a factored value function over a set $\mathcal{B} = \{\phi_1, \ldots, \phi_{|\mathcal{B}|}\}$ of Fourier boolean basis functions: $\mathcal{V}(\mathbf{x}) = \sum_j^{|\mathcal{B}|} w_j \phi_j(\mathbf{x})$. The exact Fourier representation has $2^m$ bases where $m$ is the number of state variables. However, prior work has developed an approach to select a small subset of these for a sufficiently good value function approximation [Panda and Vorobeychik, 2017].

Since the optimal policy, as well as the associated value function, is independent of interdiction strategy, we can precompute the approximate value function with an associated basis using techniques from prior art (e.g., [Guestrin *et al.*, 2003; Panda and Vorobeychik, 2017]). In the interdiction problem, the weights $w_j$ in the value function are then fixed, and the goal is to optimize over the initial state $\mathbf{x}_0$ and, consequently, the associated basis function values $\phi_j(\mathbf{x}_0)$. We now develop an integer linear program (ILP) for computing the optimal choice of the initial state $\mathbf{x}_0$ given a value function $\mathcal{V}(\mathbf{x}) = \sum_j^{|\mathcal{B}|} w_j \phi_j(\mathbf{x})$ over a pre-computed Fourier basis $\mathcal{B}$.

Let $\mathbf{b}_j$ denote a binary vector of length $m$, indicating the variables included in basis $\phi_j$. Then, for a given $\mathbf{x}_0$, $\sum_i b_{ij} x_{i0} = \sum_{i \in S_j} x_{i0}$ in the Fourier representation (2), which is an integer. Then, for some positive integer $h_j$ and a binary $k_j$, $\sum_i b_{ij} x_{i0} = 2h_j + k_j$, and is odd if $k_j = 1$ and even otherwise. If this sum is even, the corresponding basis value $\phi_j = +1$, and $\phi_j = -1$ otherwise, which we can calculate by $\phi_j = 1 - 2k_j$.

We introduce binary variables $c_{i1}$ and $c_{i2}$ for each state variable $x_i$ to compute the interdiction cost $\rho(\mathbf{x}'_0, \mathbf{x}_0)$. Putting everything together, we obtain the following ILP to compute the optimal interdiction:

$$\begin{aligned} \underset{\mathbf{x}'_0 \in \mathbf{X}, \phi, k, h, c}{\text{minimize}} \quad & \sum_{j=1}^{|\mathcal{B}|} w_j \phi_j + \sum_i \rho_i (c_{i1} + c_{i2}) \\ \text{subject to} \quad & \sum_i b_{ij} x'_{i0} = 2h_j + k_j, \forall j && \text{(3a)} \\ & \phi_j = 1 - 2k_j, \forall j && \text{(3b)} \\ & c_{i1} - c_{i2} = x'_{i0} - x_{i0}, \forall i && \text{(3c)} \\ & x'_{i0}, k_j, c_{i1}, c_{i2} \in \{0, 1\}, h_j \in \mathbb{Z}_+ \end{aligned}$$

Thus, the full interdiction approach involves two steps:

1. $[\mathcal{B}, \mathbf{w}]$ =approxMDP($\mathcal{M}$),
2. Solve ILP (3) given the approximate optimal value function $\mathcal{V}(\mathbf{x}) = \sum_j^{|\mathcal{B}|} w_j \phi_j(\mathbf{x})$.

The key bottleneck in this approach is Step 1, where the difficulty of solving the factored MDP grows exponentially in the number of interdependencies among state variables. An alternative approach is to use model-free reinforcement learning, which can scale significantly better when we use function approximation to represent the action-value function. However, this introduces new technical challenges, which we describe and address next.

# 5 Interdiction Using RL with Linear Action-Value Functions

In this section, we propose an alternative approach in which the attacker directly learns the action-value Q-function given states and actions, using a variation of the traditional Q-learning algorithm [Sutton and Barto, 1998]. Just as we used a Fourier approximation of the value function earlier, we now use this basis to approximate the optimal Q-function:

$$Q(\mathbf{x}, a; \mathbf{w}) = \sum_{j=1}^{|\mathcal{B}|} w_j^a \phi_j(\mathbf{x}). \quad (4)$$

In order to perform interdiction, we now face two new technical challenges: first, since the Fourier basis is exponential, we need to adapt the learning algorithm to iteratively construct an effective small basis representation, and second, we need to adapt the interdiction approach to work with the Q-function, rather than the value function.

We begin with the adapted Q-learning algorithm that embeds an iterative Fourier basis construction, which proceeds as follows. The attacker starts at a random state and chooses an action $a$ using an $\epsilon$-greedy strategy. The $\epsilon$ parameter decays each iteration enabling more exploitation with time. The observation $\{\mathbf{x}^t, a^t, r^t, \mathbf{x}^{t+1}\}$ at each iteration is stored in a set $\mathcal{D}$. For computational speed and higher data efficiency, we use a batch gradient descent update over a randomly sampled subset $\hat{\mathcal{D}} \subset \mathcal{D}$ collected over past iterations (similar to experience replay in [Mnih *et al.*, 2013]).

Unlike the baseline approach, addition of any new basis function is equally computationally expensive, irrespective of the interdependencies between variables in the particular basis function. We incorporate basis function selection in learning as follows. We start with an initial set $\mathcal{B} = \mathcal{B}_0$ of basis functions (e.g., all single-variable bases and the constant basis). During the learning iterations, we add a new basis function to the set $\mathcal{B}$ if it significantly reduces the squared error objective over the samples $s = (\mathbf{x}, a, r, \mathbf{x}') \in \hat{\mathcal{D}}$, $(Q'(\mathbf{x}, a; \mathbf{w}) - Q(\mathbf{x}, a; \mathbf{w}))^2$, compared to the current basis set, where $Q'(\mathbf{x}, a; \mathbf{w}) = r + \gamma \max_a Q(\mathbf{x}', a; \mathbf{w})$.

For any basis $\phi_j$ in the current basis set, the gradient descent weight update with learning rate $\eta$ for the weight $w_j^a$ of this basis (over a single observation) is:

$$w_j^a \leftarrow w_j^a + \eta(r + \gamma \max_a Q(\mathbf{x}', a; \mathbf{w}) - \sum_{j=1}^{|\mathcal{B}|} w_j^a \phi_j(\mathbf{x}))\phi_j(\mathbf{x}).$$

Now, consider a basis $\phi_n$ not present in the current approximation. In this case, the current weight $w_n^a = 0$ and the gradient update summed over the samples $s \in \hat{\mathcal{D}}$ is then

$$\mathcal{I} = \eta \sum_{s \in \hat{\mathcal{D}}} (Q'(\mathbf{x}, a; \mathbf{w}) - Q(\mathbf{x}, a; \mathbf{w}))\phi_n(\mathbf{x}).$$

Our goal is to find a basis $\phi_n(\mathbf{x})$ to add to $\mathcal{B}$ that maximizes $|\mathcal{I}|$, which measures the relative impact on the quality of the Q-function approximation. We then add this basis if $|\mathcal{I}|$ is above a predefined threshold. The following MILP computes the Boolean basis vector $\mathbf{b}$ corresponding to the new basis function with the largest marginal impact $|\mathcal{I}|$ on Q-function approximation, given samples $s^j : (\mathbf{x}^j, a^j, r^j, \mathbf{x}^{j+1}) \in \hat{\mathcal{D}}$:

$$\underset{\mathbf{b}, h, k, \phi, q, \delta}{\text{maximize}} \ \delta_1 + \delta_2$$

$$\text{subject to } \delta_1 - \delta_2 = \sum_{s_j \in \hat{\mathcal{D}}} (Q'(\mathbf{x}^j, a^j; \mathbf{w}) - Q(\mathbf{x}^j, a^j; \mathbf{w}))\phi_j \tag{5a}$$

$$0 \leq \delta_1 \leq Lq \tag{5b}$$

$$0 \leq \delta_2 \leq L(1-q) \tag{5c}$$

$$\sum_i b_i x_i^j = 2h_j + k_j, \forall \mathbf{x}^j \in \hat{\mathcal{D}} \tag{5d}$$

$$\phi_j = 1 - 2k_j, \forall \mathbf{x}^j \in \hat{\mathcal{D}} \tag{5e}$$

$$\sum_i b_i \geq 1 \tag{5f}$$

$$b_i, k_j, q \in \{0,1\}, h_j \in \mathbb{Z}_+, \delta_1, \delta_2 \geq 0.$$

$\delta_1$ and $\delta_2$ compute the linearized objective $|\mathcal{I}|$ ($L$ is a large number). Constraints (5d) and (5e) compute the $\phi_j$ values as in ILP (3). Constraint (5f) ensures that the ILP does not select the constant basis function. To keep the basis set from becoming too large, we periodically monitor the weights and remove any basis functions with normalized weights below a predefined threshold.

Given this approximation, we can now extend the interdiction LP 3 for the defender's optimal initial state $\mathbf{x}'_0$ to the following mixed integer program (recall that $\mathcal{V}(\mathbf{x}) = \max_a Q(\mathbf{x}, a; \mathbf{w})$):

$$\underset{\mathbf{x}'_0 \in \mathbf{X}, v, \phi, k, h, c}{\text{minimize}} \; v + \sum_i \rho_i (c_{i1} + c_{i2})$$

$$\text{subject to } v \geq \sum_{j=1}^{|\mathcal{B}|} w_j^a \phi_j, \forall a \tag{6a}$$

$$\sum_i b_{ij} x'_{i0} = 2h_j + k_j, \forall j \tag{6b}$$

$$\phi_j = 1 - 2k_j, \forall j \tag{6c}$$

$$c_{i1} - c_{i2} = x'_{i0} - x_{i0}, \forall i \tag{6d}$$

$$x'_{i0}, k_j, c_{i1}, c_{i2} \in \{0,1\}, h_j \in \mathbb{Z}_+$$

The full algorithm is outlined in Algorithm 1.

**Algorithm 1** Interdiction using Linear Action-Value Function Learning

Initialize weights $\mathbf{w}$ to 0, randomly initialize state, $\epsilon = \epsilon_0$
**for** iterations $t$ in $1, \ldots, T$ **do**
  Select $\epsilon$-greedy action $a^t$
  Store $s^t = (\mathbf{x}^t, a^t, r^t, \mathbf{x}^{t+1})$ in $\mathcal{D}$
  **for** each $s^j$ in a random sampled $\hat{\mathcal{D}} \subset \mathcal{D}$ **do**
    Features: $[\phi_1(\mathbf{x}^j), \ldots, \phi_{|\mathcal{B}|}(\mathbf{x}^j)]$
    Target: $r^j + \gamma \max_{a'} Q(\mathbf{x}^{j+1}, a'; \mathbf{w})$
  Gradient descent and $\mathbf{w}$ update on $\hat{\mathcal{D}}$, $\epsilon = \epsilon_0 e^{-t}$
  Every $\hat{t}$ iterations solve MILP (5), update basis set $\mathcal{B}$
Solve interdiction MILP (6) using basis set $\mathcal{B}$ and weights $\mathbf{w}$ **return** $\mathbf{x}'_0$

While this approach allows direct model-free learning and significantly more scalable interdiction (see the experiments section), the performance still relies on the subset of basis functions chosen for the approximation. We next extend the framework to allow for non-linear Q-function approximation.

# 6 Interdiction with Non-Linear Function Approximation

We now generalize the model-free interdiction framework to incorporate non-linear Q-function approximation, such as using neural network-based Q-functions. Let $Q(\mathbf{x}, a; \theta)$ denote the corresponding approximate Q-function, with parameters $\theta$ (e.g., corresponding to neural network weights).

For interdiction with a non-linear Q-function, we can no longer directly use the ILP approaches above. We instead propose two local search methods.

Our first approach is a greedy local search in the state space (Algorithm 2). The intuition behind the approach is to change one variable at a time, and accepting the change only if it improves the defender's utility. The process continues either for a fixed number of iterations, or until convergence.

**Algorithm 2** Non-Linear Value Function Learning and Greedy Local Search

Start at a randomly chosen state $\mathbf{x}'_0$
Compute $U = \max_{a'} Q(\mathbf{x}'_0, a'; \theta) + \rho(\mathbf{x}'_0, \mathbf{x}_0)$
**for** iterations in $1, \ldots, T$ **do**
  Change one state variable at random to get $\bar{\mathbf{x}}$
  **if** $\max_{a'} Q(\bar{\mathbf{x}}, a'; \theta) + \rho(\bar{\mathbf{x}}, \mathbf{x}_0) < U$ **then**
    $U = \max_{a'} Q(\bar{\mathbf{x}}, a'; \theta) + \rho(\bar{\mathbf{x}}, \mathbf{x}_0)$
    $\mathbf{x}'_0 = \bar{\mathbf{x}}$
**return** $\mathbf{x}'_0$

We observe that this local search method can be slow for a large state space. To improve efficiency of state space exploration, we propose an alternative local search approach which iteratively linearizes the Q-function using a Taylor series approximation, and solves an ILP similar to that in previous sections using the linearized function. Specifically, we start the search with a random modification of the initial state, and obtain a linear approximation of the Q-function in the vicinity of this state. The attacker's objective is now a linear function of the state. We then formulate an ILP to compute a state that minimizes this objective. The search then continues by updating the linear approximation around this local solution.

To illustrate, we derive the algorithm in the context of a neural network with a single fully connected hidden layer, and a separate output for each action. The input , hidden and output layers have $m, |H|$ and $|A|$ units respectively. The hidden layer uses a rectified linear unit activation function. The output layer has a linear activation (since it predicts the action-value function which is basically unconstrained). Let $\theta^h$ and $\theta^o$ denote the weights associated with the hidden and the output layer respectively. With the above network architecture, the decision function is given by:

$$Q(\mathbf{x}, a) = \sum_j^{|H|} \max\left[0, \sum_i^m \theta_{ij}^h x_i\right] \theta_{ja}^o \tag{7}$$

The first order Taylor series approximation of a multi-variable scalar-valued function is given by $f(\mathbf{x} + \delta\mathbf{x}) = f(\mathbf{x}) + \sum_i \frac{\partial f(\mathbf{x})}{\partial x_i} \delta x_i$. The linear approximation of the Q-function

around $\mathbf{x}$ based on our neural network architecture

$$\hat{Q}(\mathbf{x}'_0, a) = Q(\mathbf{x}, a) + \sum_i \sum_j \begin{cases} \theta^h_{ij}\theta^o_{ja}(x'_{i0} - x_i) & \text{if } \sum_i \theta^h_{ij} x_i \geq 0 \\ 0 & \text{if } \sum_i \theta^h_{ij} x_i < 0 \end{cases} \quad (8)$$

Given this local linear approximation, we can compute the optimal interdiction using the following ILP:

$$\underset{\mathbf{x}'_0 \in \mathbf{X}, v, c}{\text{minimize}}\ v + \sum_i \rho_i (c_{i1} + c_{i2})$$

$$\text{subject to } v \geq \hat{Q}(\mathbf{x}'_0, a), \forall a \quad (9a)$$

$$c_{i1} - c_{i2} = x'_{i0} - x_{i0}, \forall i \quad (9b)$$

$$x'_{i0}, c_{i1}, c_{i2} \in \{0, 1\} \quad (9c)$$

The full search algorithm is outlined in Algorithm 3.

**Algorithm 3** Interdiction with Local Linear Approximation

Start at a randomly chosen state $\mathbf{x}'_0$
**for** iterations in $1, \ldots, T$ **do**
  Compute the linear approximation as in Equation (8)
  Solve the ILP 9 to update $\mathbf{x}'_0$
**return** $\mathbf{x}'_0$

Finally, we describe two additional techniques that we used to stabilize the learning algorithm. First, at every step of training, the weights $\theta$ are updated. These constantly changing weights are used as targets in future iterations. The value estimations can thus, easily spiral out of control and destabilize learning. To alleviate this problem we use a second network to generate the target Q values and update its weights (to the primary network's weights) relatively less frequently [Lillicrap *et al.*, 2015]. Second, if the mean square error is relatively large for a sample, it can cause large changes to the network and destabilize it (the loss function is used in backpropagation for updating the weights). We use the Huber loss function to constrain this error [Mnih *et al.*, 2015].

## 7 Bayesian Interdiction Problem

Thus far, we had assumed that the interdiction game has *complete information*: both the defender and attacker know one another's utility functions and actions, as well as the initial state $\mathbf{x}_0$ that is being modified by the defender. Of course, in reality the defender is not privy to much of this information. We very generically model this uncertainty by partitioning the state variables $X$ in the MDP into three groups: $X = (X^D, X^A, X^R)$ where $X^D$ is the set of design variables that the defender can modify, $X^A$ is the set of variables known only to the attacker (but not the defender), and $X^R$ is the rest of the state variables (known to both players, but not modifiable by the defender). In particular, this uncertainty about the part of the initial state $X^A$ may capture relevant access that the attacker possesses, or which actions are available to them (for example, by having these state variables model preconditions of relevant actions).

Let $\mathcal{X}^A$ be the set of all possible attacker *types* (i.e., initial attack states). The interdiction problem can be directly extended, with the defender solving the following minimization problem:

$$\min_{\mathbf{x}^D} \left[ \mathbb{E}_{\mathbf{x}^A \in \mathcal{X}^A} \left( \mathcal{V}(\mathbf{x}^D, \mathbf{x}^A, \mathbf{x}^R) \right) + \rho(\mathbf{x}^D, \mathbf{x}_0) \right] \quad (10)$$

We can approximate this problem by replacing the expectation over a large set $\mathcal{X}^A$ with a sample average over a collection of samples of $\mathbf{x}^A$ according to the probability distribution over attacker types.

An important observation here is that the value function can be learned as a function of the *full* initial state, whether or not observed by the defender. Consequently, the proposed interdiction approaches discussed earlier can be extended directly to solve the above problem by introducing the variables and constraints specific to the sampled attacker types. In the case of neural networks, we can also extend the local search algorithms 2 and 3 by averaging over the attacker types at each search step.

## 8 Experiments

We evaluate our MDP interdiction algorithms on several instances of three problem domains from the international planning competition (IPC 2014): a) sysadmin b) academic advising and c) wildfire. While these examples have limited connection to security, they provide the most meaningful evaluation in terms of effectiveness and scalability. Previous work in security-related multi-stage attacks consider toy examples which would not provide appropriate evaluation. The most relevant prior work is Panda and Vorobeychik [2017], and offers a state-of-the-art solution to the problem. However, it considers action interdiction, rather than state interdiction. More significantly, we demonstrate that our baseline approach, which is closest to this work, has comparable running time on larger MDP instances (actually, tends to be faster). We use discount factor $\gamma = 0.9$ and set all interdiction costs $\rho_i = 1$. We denote the factored MDP interdiction as baseline (BI), and the linear and the non-linear interdiction approaches as LI, NLI1 and NLI2 respectively. We train the learning algorithms with $\epsilon_0 = 1, \eta = 0.01$ and the RMSProp optimizer for the neural networks. The batch size $|\hat{\mathcal{D}}|$ increases from $40$ to $400$ with problem size. The experiments were run on a 2.4GHz hyperthreaded 8-core Ubuntu Linux machine with 16 GB RAM, with CPLEX version 12.51 for MILP instances and TensorFlow for learning algorithms [Abadi *et al.*, 2016].

### 8.1 MDP State Interdiction

First, we compare our interdiction approaches (optimized independently) on the sysadmin domain with $m = 10 - 60$ state variables and 11 to 61 actions. Each state variable corresponds to a machine and indicates whether it is working or has failed. We consider two possibilities for the initial state $\mathbf{x}_0$: a) all machines work and b) alternate machines work. In addition, we compare against the following cases, a) no interdiction (NI): when the initial state $\mathbf{x}_0$ is not modified, and b) random interdiction (RI): when the defender modifies the initial state randomly to $\mathbf{x}'_0$. The runtime and utility (defender's gains - interdiction costs) comparisons shown in (Figure 1) demonstrate that the proposed interdiction approaches significantly improve the defender's utility compared to the baselines (recall that lower is better). The defender utility is similar for BI,LI,NLI1,NLI2 in the experiments (for a given MDP

domain). The utility differs significantly in case of NI and RI because these do not perform any optimization for interdiction and merely serve as baselines. In addition, we observe that the value function learning approaches scale much better than the baseline approach without compromising solution quality, and NLI2 tends to have the best scalability. This is primarily because these do not explicitly solve the MDP. In case of MDPs with higher order interdependencies in the transition model, scalability becomes a major bottleneck even with the approximate solution approaches with a limited number of basis functions.

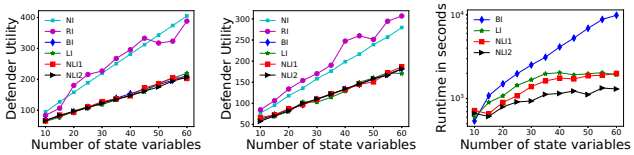


Figure 1: Comparison between the proposed interdiction approaches on the sysadmin domain in terms of utility (from two different starting states, left and center) and runtime (right).

Next, we evaluate our approaches on 10 problem instances of the academic advising domain. The problem size increases with problem number from 10 to 30 courses ($m = 20$ to 60 state variables, 10 to 30 actions). For each problem size, there are two instances, corresponding to different program requirements and course prerequisites. The first (odd numbered) problem instance is simpler (fewer prerequisites per course). The second (even numbered) instance is more complicated, with a larger number of prerequisites per course (larger number of connections in the underlying DBN). Problem 10 has the largest problem size with 30 courses, 11 program requirements, 3 prerequisites for most courses and 4 prerequisites for 8 courses. The two initial states correspond to selection of a) all courses and b) alternate courses. As demonstrated in Figure 2, we observe a similar trend as before: effectiveness of optimized interdiction and superior scalability in case of the learning-based approaches.

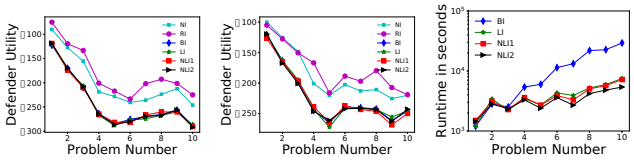


Figure 2: Comparison between the proposed interdiction approaches on the academic advising domain in terms of utility (different starting states, left and center) and runtime (right).

Finally, we evaluate on 6 problem instances of the wildfire domain. The problem is defined on a grid and the size increases with problem number from $n = 3$ to 5 ($m = 2 \times n^2 = 18$ to 50 state variables, 36 to 100 actions). For each grid size, there are two instances, corresponding to different neighbourhood configurations and targets (cells on the grid that need to be protected). The first (odd numbered) problem instance has fewer targets than the second (even numbered) instance. In this case, we scale down the original rewards by a factor of 100 to ensure better convergence of the learning algorithms. The results are shown in Figure 3, and are broadly consistent with previous observations.

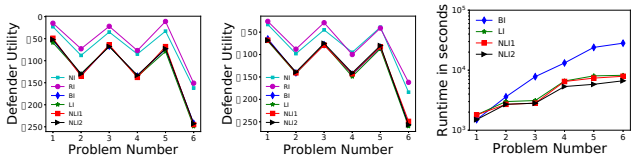


Figure 3: Comparison between the proposed interdiction approaches on the wildfire domain in terms of utility (different starting states, left and center) and runtime (right).

## 8.2 Bayesian Interdiction

Our final set of experiments deals with Bayesian interdiction. As a baseline, we consider interdiction of a worst-case attack. We obtain the initial state corresponding to this attacker by maximizing the approximate value function. The defender's problem is then given by $\min_{\mathbf{x}^D}[\max \mathcal{V}(\mathbf{x}^D, \mathbf{x}^A, \mathbf{x}^R) + \rho(\mathbf{x}^D, \mathbf{x}_0)]$.

In each of the domain examples considered, we divide the state variables as $X = (X^D, X^A, X^R)$, with $40\%$, $40\%$ and $20\%$ relative proportions. We randomly sample $500$ assignments to the attacker's variables, with equal probability. In each case, we plot the difference in the defender's utility, between the baseline and the Bayesian cases. The results in the Figures 4, 5 and 6 exhibit a large *decrease in utility* (of the attacker), that is, a large benefit to the defender from considering Bayesian, rather than baseline, interdiction. While the runtime does increase somewhat, this increase is small compared to the time it takes to solve the MDP.

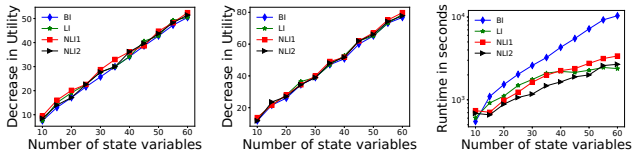


Figure 4: Improvement in the defender's utility using Bayesian interdiction in the sysadmin domain (different starting states, left and center) and interdiction runtime (right).

# 9 Conclusions

We presented a novel interdiction model in which the defender constrains the initial state of the attacker. We proposed scalable interdiction techniques with single-level integer linear programming, compared to difficult bi-level problems discussed in previous work. We further improved scalability by using model-free reinforcement learning techniques with linear and non-linear action-value function approximators. We extended the interdiction framework to Bayesian interdiction.

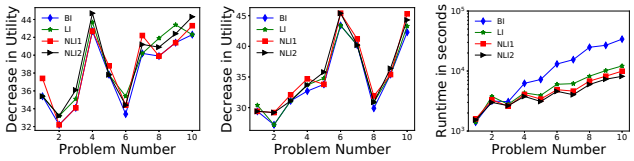


Figure 5: Improvement in the defender's utility using Bayesian interdiction in the academic advising domain (different starting states, left and center) and interdiction runtime (right).

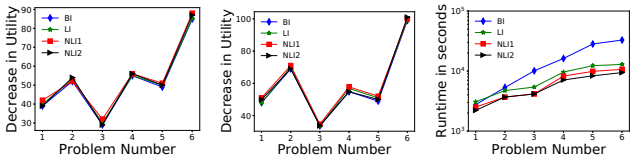


Figure 6: Improvement in the defender's utility using Bayesian interdiction in the wildfire domain (different starting states, left and center) and interdiction runtime (right).

Finally, we evaluated the effectiveness of our proposed approaches on several realistic MDP problem instances.

## Acknowledgments

This research was partially supported by the National Science Foundation (CNS-1640624, IIS-1526860, IIS-1649972), Office of Naval Research (N00014-15-1-2621), Army Research Office (W911NF-16-1-0069), and National Institutes of Health (UH2 CA203708-01, R01HG006844-05).

## References

[Abadi *et al.*, 2016] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[Ammann *et al.*, 2002] Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 217–224. ACM, 2002.

[Boddy *et al.*, 2005] Mark Boddy, Johnathan Gohde, Tom Haigh, and Steven Harp. Course of action generation for cyber security using classical planning. In *International Conference on Automated Planning and Scheduling*, pages 12–21, 2005.

[Guestrin *et al.*, 2003] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

[Jain *et al.*, 2008] Manish Jain, James Pita, Milind Tambe, Fernando Ordónez, Praveen Paruchuri, and Sarit Kraus. Bayesian stackelberg games and their application for security at los angeles international airport. *ACM SIGecom Exchanges*, 7(2):10, 2008.

[Koller and Parr, 1999] Daphne Koller and Ronald Parr. Computing factored value functions for policies in structured mdps. In *IJCAI*, volume 99, pages 1332–1339, 1999.

[Korzhyk *et al.*, 2011] Dmytro Korzhyk, Zhengyu Yin, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe. Stackelberg vs. nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *J. Artif. Intell. Res.(JAIR)*, 41:297–327, 2011.

[Krautsevich *et al.*, 2012] Leanid Krautsevich, Fabio Martinelli, and Artsiom Yautsiukhin. Towards modelling adaptive attacker's behaviour. In *International Symposium on Foundations and Practice of Security*, pages 357–364. Springer, 2012.

[Letchford and Vorobeychik, 2013] Joshua Letchford and Yevgeniy Vorobeychik. Optimal interdiction of attack plans. In *International Conference on Autonomous Agents and Multi-Agent Systems*, pages 199–206, 2013.

[Lillicrap *et al.*, 2015] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[Mnih *et al.*, 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[Ng *et al.*, 2010] Brenda Ng, Carol Meyers, Kofi Boakye, and John J Nitao. Towards applying interactive pomdps to real-world adversary modeling. In *IAAI*, 2010.

[Obes *et al.*, 2013] Jorge Lucangeli Obes, Carlos Sarraute, and Gerardo Richarte. Attack planning in the real world. *arXiv preprint arXiv:1306.4044*, 2013.

[O'Donnell, 2008] Ryan O'Donnell. Some topics in analysis of boolean functions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 569–578. ACM, 2008.

[Panda and Vorobeychik, 2017] Swetasudha Panda and Yevgeniy Vorobeychik. Near-optimal interdiction of factored mdps. In *Conference on Uncertainty in Artificial Intelligence*, 2017.

[Paruchuri *et al.*, 2008] Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 895–902. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[Puterman, 1994] Martin L Puterman. Markov decision processes: Discrete stochastic dynamic programming. 1994.

[Sutton and Barto, 1998] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[Watkins and Dayan, 1992] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.