Solving Online Threat Screening Games using Constrained Action Space Reinforcement Learning

Sanket Shah,[1] **Arunesh Sinha,**[1] **Pradeep Varakantham,**[1] **Andrew Perrault,**[2] **Milind Tambe**[2]
[1]School of Information Systems, Singapore Management University, {sankets, aruneshs, pradeepv}@smu.edu.sg
[2]Center for Research on Computation and Society, Harvard University, aperrault@g.harvard.edu, milind_tambe@harvard.edu

Abstract

Large-scale screening for potential threats with limited resources and capacity for screening is a problem of interest at airports, seaports, and other ports of entry. Adversaries can observe screening procedures and arrive at a time when there will be gaps in screening due to limited resource capacities. To capture this game between ports and adversaries, this problem has been previously represented as a Stackelberg game, referred to as a Threat Screening Game (TSG). Given the significant complexity associated with solving TSGs and uncertainty in arrivals of customers, existing work has assumed that screenees arrive and are allocated security resources at the beginning of the time-window. In practice, screenees such as airport passengers arrive in bursts correlated with flight time and are not bound by fixed time-windows. To address this, we propose an online threat screening model in which the screening strategy is determined adaptively as a passenger arrives while satisfying a hard bound on acceptable risk of not screening a threat. To solve the online problem, we first reformulate it as a Markov Decision Process (MDP) in which the hard bound on risk translates to a constraint on the action space and then solve the resultant MDP using Deep Reinforcement Learning (DRL). To this end, we provide a novel way to efficiently enforce linear inequality constraints on the action output in DRL. We show that our solution allows us to significantly reduce screenee wait time without compromising on the risk.

1 Introduction

Screening for potential threats entering large safety-sensitive establishments (e.g., airports, seaports, museums) using the right subset of available screening methods (e.g., metal detectors, advanced imaging technology, pat-down) is an important defensive activity undertaken by various agencies around the world. However, the sheer scale of the problem at these large establishments with a large number of screenees and various screening methods makes screening in a timely fashion with limited resources quite challenging. For example, average delays for air passengers in Chicago, USA jumped to 2 hours due to higher passenger volume in the summer of 2016 (Dastin 2016). Additionally, intelligent adversaries can exploit any gaps in screening and cause catastrophic damage to these critical establishments. Screening gaps can arise due to the use of a less effective, but faster, screening method for high-risk passengers, which can happen as the combination of more passengers and limited resources can result in the unavailability of the "right" screening method.

Online resource allocation is a problem of interest in many domains including transportation (Simao et al. 2009) – allocating taxis to customers, emergency response (Maxwell et al. 2010) – allocating ambulances to emergencies, and airports – allocating terminals to arriving aeroplanes. Threat screening is also an online resource allocation problem, but in the presence of an observing adversary. Given that adversaries can monitor resource allocation strategies and exploit any gaps in them, game-theoretic models and approaches have been used to consider robust or risk-averse objectives rather than traditional expected objectives (e.g., expected revenue, expected delay).

One such model is the Threat Screening Game (TSG) model introduced in Brown et al. (2016). Here, a strategic attacker attempts to enter a secure area while the defender uses teams of limited-capacity screening resources (with varying efficacies) to thwart them. However, despite enhancements made in subsequent work (McCarthy, Vayanos, and Tambe 2017), this model suffers from a lack of adaptability as screening strategies are fixed for every hour. Further, all versions of the model assume a favourable rate of passenger arrival within each time-window such that no screening resource is idle within the hour. We address these shortcomings with a novel online allocation model and a completely new solution approach.

Our *first* contribution is an online version of the threat screening problem, in which the screening strategy is decided adaptively, based on the current queue lengths, as the screenees arrive. We show experimentally that this leads to a much better characterization and optimization of the average delay time faced by screenees at no loss to security risk (measured as attacker utility) compared to past work. Further, while past models have used a weight to balance the risk of missing an attacker and average delay time, we

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

impose a hard bound on risk while simultaneously minimizing delay. We show that, given uncertain and unknown passenger arrivals, the online model can be solved as a Reinforcement Learning (RL) problem with continuous action space where the hard bound on risk translates to hard constraints on the action space. We show mathematically that the choice of hard bound on risk is not different from the one where a weighted defender objective is maximized and we can switch back and forth between these two seemingly different optimization goals. Our mathematical analysis also reveals the game-theoretic nature of this formulation.

Our *second* contribution is a novel method to efficiently impose hard constraints on actions in Deep RL by using what we call α-projection. In contrast to prior approaches (see Section 2), our approach guarantees that the constraint is never violated (even during training) while also being much more scalable in training as well as execution. The main component of our method is an extremely efficient mapping of infeasible actions to the feasible space specified by the constraints.

Finally, our *third* contribution is a set of experiments that reveal why and how prior TSG models fail to handle realistic continuous arrival of passengers in bursts. The experiments also show that our approach achieves the same risk as prior models but improves upon the average delay by 100% in the best case and 25% on average.

Overall, the realism of our model coupled with a novel scalable RL solution method makes our approach appealing for practical large scale threat screening problems.

2 Related Work

TSG and Security Games

There have been quite a few papers published on various aspects of threat screening games. The early papers (Brown et al. 2016; Schlenker et al. 2017) make two stringent assumptions – first, they assume perfect prior knowledge of passenger arrivals (for one hour time-windows) and second, they implicitly handle delay by assuming that all passengers are screened within the same window in which they arrive (thus, delay is not an explicit consideration in this work). Both these assumptions are unrealistic in practice as, clearly, there is uncertainty in the number of passengers arriving in any time-window and passengers arrive in bursts that are correlated with the flight timings, which can cause large average delay. For example, if a screening resource can screen one passenger every 6 minutes, prior work assumed that 10 passengers can be screened with this resource in an hour irrespective of when they arrived. Clearly, if all the passengers arrive in the last 15 min of the hour, they can't all be screened, or if passengers arrive in a burst together, the average waiting time will be higher than when they arrive equally spaced apart every 6 minutes. We show in our experiments how these assumptions result in sub-optimal outcomes in practice.

Later papers (McCarthy, Vayanos, and Tambe 2017; McCarthy et al. 2018) attempt to account for uncertainty in arrivals across time-windows and relax the second assumption by allowing for an overflow of passengers from one time-window to the next. However, their approach is unscalable and, hence, impractical for real-world application. They show solutions for only up to 15 flights for a full day. Additionally, because the solution goal chosen is inspired from robust optimization, the method tries to find a solution that minimizes risk and delay across any realizable sample, which results in a pessimistic solution. Further, the solution approach also makes the approximation of calculating the worst case from a sample of arrivals and, as a result, cannot guarantee that the solution will bound the true worst-case security risk. Finally, this work continues to handle delay implicitly by assigning the same overall delay for passengers that are to be screened in one time-window irrespective of when they arrive. Thus, a long delay for many passengers, when all of them arrive at the start of the hour, is considered same as when passengers arrive spaced equally apart in time with no delay for each passenger. Such a coarse measure of delay results in sub-optimal outcomes in terms of delay, as we show in experiments.

In this paper, we propose a scalable online model without the restrictive assumptions of past work: we guarantee a bound on the worst-case risk and simultaneously minimize the average delay. Our online model allows for fine-grained adaptivity at the level of individual passenger arrivals as opposed to the hourly time-window adaptations in past work and also scales up to a large number of flights.

In other applications of security games played over multiple time steps, there has been work in which the defender strategy is a policy for an MDP or a sequence of actions (Delle Fave et al. 2014; Bosansky et al. 2015). This work assumes that the MDP or game parameters are known beforehand and is hence a planning problem rather than a learning problem. Additionally, there has been work in this space where the double oracle approach has been used in tandem with Deep RL to compute the equilibrium (Wang et al. 2019; Wright, Wang, and Wellman 2019) or fictitious play based policy gradient Deep RL has been used to compute equilibrium (Kamra et al. 2018; 2019). In these prior work, however, there is either no constraints on the action or only a single constraint on actions (actions sum to one), which is readily enforced using a softmax layer. In this paper, our main contribution to Deep RL is in enforcing multiple arbitrary linear inequality constraints efficiently. There is also theoretical work on solving security games in an extensive form (Letchford and Conitzer 2010; Kroer, Farina, and Sandholm 2018; Černý, Boỳanskỳ, and Kiekintveld 2018; Basilico, Gatti, and Amigoni 2009) or stochastic game (Letchford et al. 2012) setting. Again, these assume complete knowledge of the game structure including transition functions whereas we focus on learning aspects. Also, whereas learning in the context of security games has appeared in the literature (Balcan et al. 2015; Letchford, Conitzer, and Munagala 2009), these methods do not directly apply to our RL problem formulation.

Constrained Action-Space RL

Historically, dealing with constraints on the action space in Deep RL has been a challenging task. This is exacerbated by the fact that our action space is continuous. The most

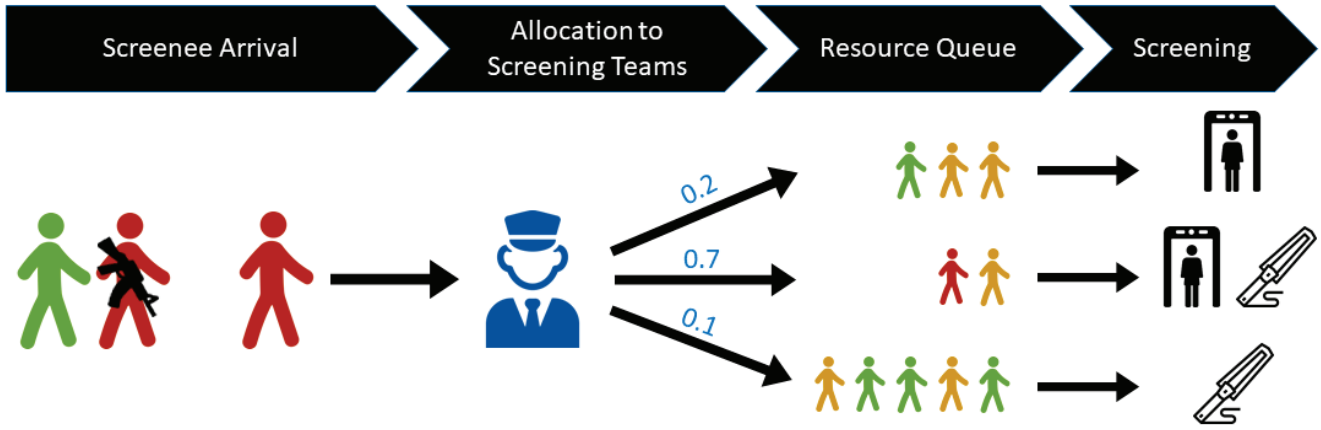

Figure 1: An overview of the screening process. First, a screenee of a given category (differentiated by colour) arrives for screening. On arrival, they are immediately assigned to one of the security teams according to a stochastic policy. The probabilities of being assigned to each team is denoted by the numbers in blue. Once assigned, they wait in line to get screened. If they pass the screening, they are let into the protected area. The challenge in the TSG is in how to assign screenees to teams so that both the security risk and delay are minimised.

common and intuitive technique is to discourage disallowed actions with a penalty. This method does not guarantee that the security risk will be bounded, however, and given that the risk is the worst-case allocation across an episode, the probability and extent of violation increases with scale. Given the adversarial nature of the security problem, this method is unsuitable.

Recently, Pham, De Magistris, and Tachibana (2018) have suggested enforcing constraints by projecting any unconstrained point onto the constrained space by solving an optimisation program that minimises the L2 distance and back-propagating through it to train the network (Amos and Kolter 2017). This approach is very time consuming as it requires solving a quadratic program (QP) in the forward pass in every training iteration and, as a result, does not scale to problems with large dimensional action spaces (Amos and Kolter 2017) seen in practical screening problems.

Our RL approach is similar in spirit to Bhatia, Varakantham, and Kumar (2019), which uses a complicated variable-length iterative approximation of the L2 projection to deal with a specific subset of linear constraints faster than Pham, De Magistris, and Tachibana (2018). The type of linear constraints they can handle are constraints on the sum of sets of variables, where these sets must form a hierarchy. In contrast, the approach that we propose can handle arbitrary linear constraints, that is, in which the coefficients take any real values. Moreover, our approach is simple, can be computed in a single step, and is easy to implement as gradients can be computed using automated symbolic differentiation.

It is worth noting here that there has been significant recent work under the umbrella of "Constrained Reinforcement Learning" Achiam et al.; Chow et al. (2017; 2017). Those methods try to solve a fundamentally different problem from the one considered in this paper, however. There, the constraints are placed on an *expected long-term* secondary cost (e.g., ensuring that a mars rover lands by a certain deadline) while maximizing a primary long-term reward (e.g., landing the rover safely). In this work, we are interested in the paradigm of "Action Constrained RL" in which the constraints are enforced on an *immediate* secondary cost (e.g., violating security risk or not) that has to hold even in the *worst case*. Since constraint enforcement on long-term expected costs cannot guarantee constraint enforcement on immediate worst-case costs, these prior methods cannot be employed in our problem.

3 Threat Screening Games (TSGs)

An overview of the screening procedure considered in Threat Screening Games can be found in Figure 1. While we consider an online version of the problem in this paper, the basic structure stays the same as prior TSG models.

Every arriving passenger has a category $c \in C$, which is made up of two parts $\langle \theta, \kappa \rangle$, where θ is the part of the category that the attacker cannot control (risk level determined by screener) and κ is the part that they can control (which flight to take). The screening resource types comprise the set R; for example, R could be {X-Ray, Metal Detector, Advanced imaging}. Each resource type $r \in R$ has a rate of screening (called capacity in prior work) given by f_r measured in units of passengers per unit time. The passengers are screened by a team (set) of screening resources where the set of all teams T is given a priori ($T \subset 2^R$). An attacker, apart from choosing a flight, uses an attack method m (e.g., knife or gun) and each team t has an effectiveness (probability) $E_{t,m}$ of detecting attack method m. $U^+_{\kappa,m}$ is the defender's utility for detecting an attacker with attacker's choice being κ, m and $U^-_{\kappa,m}$ is the utility of not doing so.

As in previous TSG models, the adversary's utilities are the negation of these values. The defender has a belief about the attacker's uncontrollable category θ given by P_θ with $\sum_\theta P_\theta = 1$. Additionally, as in past work (McCarthy, Vayanos, and Tambe 2017), we work in the marginal space

in this paper, i.e., we represent the randomized allocation as the marginal probability of allocating each screenee, instead of a probability distribution over an exponentially large set of integral allocations.

4 MDP Model of TSGs

In this paper, we model the online nature of screening by formulating the TSG as a Markov Decision Process (MDP). By defining a state as the arrival of a new screenee, we determine the screening strategy for passengers (actions) as they arrive. This is unlike past approaches in which the same randomized screening strategy was used for every passenger of a given category that arrived in the same time-window.

The crux of our approach lies in the insight that we can convert an upper bound on a best responding adversary's payoff in TSGs into a constraint on the screening policy. As we specify below, one of the terms in the defender's utility (which we call detection utility) is the negation of the adversary's utility, thus, to ensure that the adversary will never have utility more than some ψ, all we have to do is to make sure that the defender chooses actions that ensures more than $-\psi$ detection utility to itself. Concretely, this constraint on the defender policy boils down to a constraint on each action (randomized assignment) in every time step – as long as the defender never makes any randomized assignment that has more than ψ expected utility for the attacker, the overall policy will never have more than ψ utility for the attacker. Our MDP formulation is parameterized by this bound ψ on the attacker utility (security risk). after presenting our model we show how the defender policy learned for the varying parameter ψ can be leveraged to compute an equilibrium in the game-theoretic sense.

The MDP treatment also considerably simplifies the problem of dealing with passenger arrival uncertainty by incorporating those in the MDP transition model. We describe our MDP formulation in detail below:

- **States:** The state at any given point in time is a combination of 4 quantities $\langle c, \xi, h, \tau \rangle$. The first, c, is the category of the passenger that has arrived for screening and we have to allocate security resources to. The remaining quantities summarise the history and provide information about the current context. $\xi \in \mathbb{R}^{|R|}$ encodes the number of passengers (or part thereof) in the queue for each resource at this current point in time. $h \in {\mathbb{Z}_+}^{|C|}$ is a summary of the history: it is the number of passengers from every category that have already been screened. τ is the wall clock time when the passenger arrives.

- **Actions:** An action $\pi_t \in \mathbb{R}^{|T|}$ at time step t is a randomized allocation of the just-arrived passenger to teams. We use the insight that the risk is a function of the policy and does not depend on passenger arrivals to codify the hard bound as risk as constraints on the action space. Only actions with risk less than the specified risk level are allowed. Risk in TSGs is measured as the expected adversary utility, which is the negation of the utility of the defender. This leads to m different inequalities constraints (one for each attack method) on the action stated in terms of defender utility.

$$P_\theta * \left[z_m U^+_{\kappa,m} + (1 - z_m) U^-_{\kappa,m} \right] \geq -\psi_\theta \quad \forall m, \quad (1)$$
$$\sum_{t \in T} E_{t,m} \pi_t = z_m \quad \forall m, \text{ and } \sum_{t \in T} \pi_t = 1$$

Given a marginal policy π, z_m is the overall probability that one of the teams $t \in T$ will detect an attack of type m. The last equality constraint enforces that π_t is a probability distribution over teams. In order to explain the first set of inequalities, we first state the defender detection utility explicitly. $U_{\kappa,m} = z_m U^+_{\kappa,m} + (1 - z_m) U^-_{\kappa,m}$ is the expected utility of the defender if the current passenger is an attacker. The utility $U_\theta = \min_{\kappa,m} U_{\kappa,m}$ is the worst case expected utility when the attacker is one with uncontrollable category θ. $\sum_\theta P_\theta U_\theta$ is the overall defender detection expected utility. We wish to impose a lower bound $-\psi_\theta$ on $P_\theta U_\theta$ which indirectly lower bounds $\sum_\theta P_\theta U_\theta$ (or in other words, upper bounds risk). It is easy to see that the first set of inequalities is $P_\theta * (\min_m U_{\kappa,m}) \geq -\psi_\theta$. Since this inequality is applied for every passenger with uncontrollable category θ, we get $P_\theta * (\min_{\kappa,m} U_{\kappa,m}) \geq -\psi_\theta$, which is nothing but $P_\theta U_\theta \geq -\psi_\theta$. Thus, this guarantees that the overall defender detection expected utility $\sum_\theta P_\theta U_\theta \geq -\sum_\theta \psi_\theta$ (or risk is bounded from above by $\sum_\theta \psi_\theta$). As these inequalities hold for any choice of action by attacker, this guarantees $-\sum_\theta \psi_\theta$ detection utility against a best responding attacker.

- **Transitions:** The transition from $\langle c, \xi, h, \tau \rangle$ to $\langle c', \xi', h', \tau' \rangle$ can be decomposed into three parts. The first is how the allocation at the previous step and passage of wall clock time since then affects the queues for each resource. Passengers in the resources queues are screened according to the screening rate of a given resource:

$$\xi'_r = \max(\xi_r - (\tau' - \tau) * f_r, 0) \text{ for all } r.$$

The second part controls h:

$$h'_c = h_c + 1, h'_d = h_c \text{ for all } d \neq c.$$

The final part is determined by passenger arrivals and represents the likelihood of arrival of a passenger of a given type at a given time $P(c', \tau' | h, c, t)$. This is a function of the arrival history, but is unknown, which motivates our use of RL for the problem.

- **Rewards:** The reward for each time step t is the negative of the expected wait time of the currently arrived passenger. The wait time is determined by the maximum wait time over all resources in the realized team allocation. The wait time for each resource r is determined by the screening rate f_r and the number of passengers ξ_r already in queue for that resource: ξ_r / f_r. We use $U_{o,t}$ to denote the delay reward at time t (note $U_{o,t}$ is negative). The value (long term reward) is given by $V_o = \mathbb{E}[(1/N) \sum_{t=1}^{N} U_{o,t}]$, where N passengers arrive in a day (implicitly conditional on the start state with empty history).

Relationship to Game Theory

In the above constrained RL problem, the defender learns a policy which is a mixed strategy of the defender (mixed since the allocation at each time step is randomized). The adversary observes this policy and chooses an optimal attack a, which is a combination of κ and attack method m. Thus, this is a Stackelberg game setting, similar to prior models of TSG. There are two components of the defender's value function: (a) the risk of not detecting the adversary, captured in $\sum_\theta P_\theta U_\theta$ and (b) the effect of delay, captured in V_o. The above RL approach solves the following problem $\max_{\pi \in \mathcal{F}_\psi} V_o(\pi)$ where π represents policies and $\mathcal{F}_\psi = \{\pi \mid P_\theta U_\theta(\pi_t, a) \geq -\psi_\theta$ for all attacker actions a and all $\theta\}$. Observe that here we explicitly write the arguments for U_θ and V_o. In particular, V_o does not depend on the attacker action and the definition of $\mathcal{F}_\psi$ ensures achieving a minimum of $-\sum_\theta \psi_\theta$ detection utility against a *best responding* adversary.

While the RL approach restricts the policy space of the defender via a bound on risk, one may wonder if the defender can achieve higher utility without such a restriction. Another way to view the problem is where the defender optimizes $\sum_\theta P_\theta U_\theta + w * V_o$ over all possible π without any restrictions, where w is a constant weight that specifies the relative importance of minimizing risk and average delay time of passengers. While our approach requires the defence agencies to specify acceptable security risk level, this other approach requires specifying a trade-off weight w between two completely different types of utilities (security risk and delay), which is why we believe that the hard bound on security risk based model is more natural. In any case, we show a relation between these two approaches that allows us to switch back and forth between them.

Theorem 1. *There exists a ψ (dependent on w) such that any $\pi^* \in \operatorname{argmax}_{\pi \in \mathcal{F}_\psi} V_o(\pi)$ is the defender strategy part of a Strong Stackelberg Equilibrium (SSE) of the Stackelberg game defined with defender objective as $\sum_\theta P_\theta U_\theta + w * V_o$.*

Proof. An SSE is one which maximizes $\sum_\theta P_\theta U_\theta + w * V_o$, subject to the best response of the attacker. The definition $U_\theta = \min_{\kappa,m} U_{\kappa,m}$ already takes care of the best response of the attacker as the attacker utility is $-U_{\kappa,m}$ and the attacker action κ, m that minimizes $U_{\kappa,m}$ maximizes $-U_{\kappa,m}$. Also, as $U_{\kappa,m}$ is continuous in π and min of continuous functions is continuous, $\sum_\theta P_\theta U_\theta$ is continuous in π.

The space of possible π is compact, thus, the continuous bounded function $\sum_\theta P_\theta U_\theta + w * V_o$ of π achieves a maximum at some $\pi^* \in \Pi^*$. This set Π^* is the set of defender strategies that form an SSE of the game. Let the value U_θ^* and V_o^* be obtained at this π^*.

Consider the values $-\psi_\theta^* = P_\theta U_\theta^*$ and $\psi^* = \langle \psi_\theta^* \rangle_{\theta \in \Theta}$. Note that $\mathcal{F}_\psi$ is specified by linear inequalities given by Equation 1, thus, $\mathcal{F}_\psi$ is a polytope. Also, $\pi^* \in \mathcal{F}_{\psi^*}$. We claim that the optimal solution of $\max_{\pi \in \mathcal{F}_{\psi^*}} V_o(\pi)$ is in $\Pi^* \cap \mathcal{F}_{\psi^*}$, which is not empty as $\pi^* \in \Pi^* \cap \mathcal{F}_{\psi^*}$. As $\sum_\theta P_\theta U_\theta^* + w * V_o^*$ is the global maximum, for any $\pi \in \mathcal{F}_{\psi^*}$ if $\sum_\theta P_\theta U_\theta + V_o = \sum_\theta P_\theta U_\theta^* + V_o^*$ then $\pi \in \Pi^*$. And also, there does not exist any $\pi \in \mathcal{F}_{\psi^*}$ such that $\sum_\theta P_\theta U_\theta + V_o > \sum_\theta P_\theta U_\theta^* + V_o^*$, which proves our claim. Since the optimal solution is in Π^*, this proves our result for ψ^*. □

The above theorem also provides an easy algorithm to solve for an approximate SSE in the unrestricted game using the RL approach. The approach is to construct a Pareto frontier a priori by solving for the optimal policy for many values of ψ where these values are uniformly spaced and distributed throughout the possible space of ψ values. Then, when given w, the solution will choose one of the specific points for which the output π maximizes $\sum_\theta P_\theta U_\theta + w * V_o$ over all points considered in the ψ space.

5 Overall Solution

To solve the screening problem modelled in Section 4, we use Reinforcement Learning (RL). We use techniques from RL instead of trying to solve the MDP directly because the exact passenger arrival distribution is unknown. Rather than trying to model the distribution explicitly, we use model-free RL techniques to jointly learn the distribution and the optimal policy.

Specifically, we use the Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al. 2015) algorithm that is a state-of-the-art technique in Deep Reinforcement Learning literature. DDPG is an extension of the standard actor-critic approach that allows the modelling of a continuous action space like the one present in our problem. However, DDPG cannot enforce general action-space constraints as is.

To deal with this, we propose an α-projection layer in Section 6 that enforces constraints on the output of the previous layer. We then modify the standard DDPG algorithm by adding this α-projection layer on top of the output layer of the actor network. This ensures that any action produced by the actor satisfies the risk constraints on the action space. This combination of DDPG and α-projection represents the overall approach that what we use to solve the MDP.

6 Enforcing Linear Constraints on Continuous Action Spaces in Deep RL

We wish to enforce the security risk constraints on the actions at every time step in our problem. Also, these security risk constraints *must* always be enforced, that is, these should never be violated.

All prior approaches for imposing *hard* constraints on the action output of any policy neural network use a layer(s) at the end of the network to map the unconstrained output from intermediate layers to an output in the feasible space. Mathematically, suppose the output must lie in a feasible space Y defined by linear inequality constraints. Let $f(x)$ be the output of the intermediate layer, given input x. The last layer(s) define a mapping M such that $M(f(x))$ lies in the feasible space Y. For our problem, Y is a fixed polytope for a given problem instance (see Equation 1). Typically, such mappings M have been some type of L_p projection ($p = 1$ or 2) in the past. This projection is written as an optimization problem and enforced as a neural network layer using techniques such as OptLayer (Pham, De Magistris, and Tachibana 2018). However, such mappings are

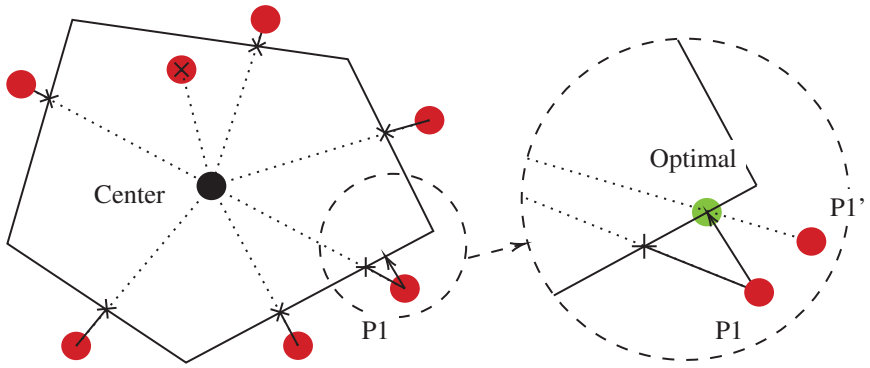

Figure 2: A schematic representation of α-projection. *Left*: The feasible region is shown as the polygon. The red points are produced by an intermediate layer. The α-projections of the red points are shown with the cross marks. If the red point is inside the feasible region, the α-projection of the point is the point itself. For one red point (inside the small dashed circle), the L2 projection is shown with the arrow. **Right**: A zoomed-in version of the dashed circle. This shows that if the L2 projection of P1 is the optimal point (green), then the intermediate layer will adjust its output to P1' to get the optimal point with the α-projection.

expensive to compute in practice as they require solving a quadratic program for every training iteration and every execution. This creates the need for an efficient mapping. First, we list desirable properties of such a mapping.

- *Onto:* To make sure that the neural network has the opportunity to output any value in the feasible space. This ensures that there is no loss in solution quality arising from a restricted solution space.
- *Continuous everywhere and differentiable almost everywhere:* This allows the neural network to learn through this mapping layer by backpropagating gradients.[1]
- *No vanishing or exploding gradients:* The gradients of the mapping should be informative when optimizing loss, that is, should not be zero or very large for many points.

Observe that the mapping M need not be a closest point (in any distance) in Y to the unconstrained $f(x)$. This is because the whole neural network is the function composition $M \circ f$ and the training ensures that the output $M \circ f$ minimizes the loss. The purpose of M is to only ensure feasibility of output, thus, any choice of M (with the properties above) will work since the neural network will appropriately adjust f so that $M \circ f$ is optimal.

α-projection

We propose a very efficient mapping M, which we call α-projection. We start by finding a feasible point $y_0 \in int(Y)$, where $int(Y)$ are all the interior points of Y. Then, we find the maximum $\alpha \in [0, 1]$ such that $y = \alpha * f(x) + (1 - \alpha) * y_0$ and $y \in Y$, with the final output being the point y, so $M(f(x)) = y$. Intuitively, we join $f(x)$ and y_0 with a line and choose the closest point to $f(x)$ on this line that lies in Y, which could be $f(x)$ itself if $f(x) \in Y$. See Figure 2.

Forward pass α-projection turns out to be very efficient for linear constraints such as those in the TSG-RL problem. For training neural networks, every iteration requires a forward pass for the network also. An optimization layer (such as our α-projection layer) requires solving an optimization problem. However, our simple mapping allows obtaining α in a closed-form with no need to solve expensive optimizations. To get to the closed-form, first, lets consider m inequality constraint with the i^{th} constraint being $a_i \cdot y \leq b_i$. Using our mapping, this i^{th} constraint is $\alpha * (a_i \cdot f(x)) + (1 - \alpha) * (a_i \cdot y_0) \leq b_i$. Thus, $\alpha * (a_i \cdot f(x) - a_i \cdot y_0) \leq b - a_i \cdot y_0$. We get a similar upper (or lower, depending on the sign of $a_i \cdot f(x) - a_i \cdot y_0$) bound for α for every inequality constraint and then α is the simply the highest value in $[0, 1]$ that satisfies all these bounds. All these bounds are closed-form formulas, thus, computing α is very efficient for the forward pass, and obtaining $y = \alpha * f(x) + (1 - \alpha) * y_0$ is easy.

Gradients: Computing the gradient (for backpropagation) of α w.r.t. the input $f(x)$ to the mapping layer is also easy. For readability, we write s instead of $f(x)$. As stated in the previous paragraph α is the minimum of a number of upper bounds (ignroing lower bounds), where these upper bounds are given closed-form functions $b_i(s)$ (y_0 is a constant). Thus, $\alpha = \min(1, b_1(s), \ldots, b_k(s))$ for some $k < m$. For notational ease, let $b_0(s) = 1$. For any specific s_0, if there is a unique index j for which $\alpha = b_j(s)$ then the gradient is simply $\nabla b_j(s)$, which is 0 if $j = 0$. If there is a set of indices J ($|J| > 1$) and $\alpha = b_j(s)$ for all $j \in J$, then the gradient is simply $(1/|J|) * (\sum_{j \in J} \nabla b_j(s))$. In practice, these gradients need not be explicitly calculated and can be handled by automatic symbolic differentiation libraries (Abadi et al. 2015) instead. Thus, our approach is also simple to implement.

Handling equality constraints: For equality constraints, say k constraints, the general approach would be to eliminate k variables using Gaussian elimination (or any other method) and then deal with the only inequalities in this new space. However, for our TSG problem, we only have one

[1]Non-differentiability for points in a measure zero set are allowed, as is in the ReLU activation.

equality constraint, which is a probability simplex constraint that can be easily enforced by a softmax layer. With a slight abuse of notation, we use s to denote the output of the softmax whose input is the unconstrained output $f(x)$. s satisfies the probability simplex constraint. Additionally, we choose y_0 such that it also satisfies the probability simplex constraint. Then, the final output $\alpha * s + (1-\alpha) * y_0$ satisfies the probability simplex constraint and all the inequality constraints. Thus, for our problem, the overall mapping is made of 2 layers: a softmax followed by the α-projection layer.

Choosing an Interior Point

The choice of y_0 is important. First, y_0 should be an interior point. Otherwise, if y_0 is on an external face (hyperplane) of the Y polytope, all external points on the side of the face not containing the polytope will map to y_0. This violates non-zero gradients property of a feasible mapping, as all points on one side of the hyperplane will have zero gradients. We also find that we get better performance when y_0 is near the centre of the polytope. While there are many different types of centres, we choose the Chebyshev centre of a polytope because: (1) Chebyshev centre can be computed efficiently by solving a linear program and (b) the Chebyshev centre maximizes the minimum distance from the faces of the polytope, that is, making sure the centre is far from bad points. Informally, Chebyshev centre is the centre of the largest ball that fits inside the polytope. Note that we need to compute the Chebyshev centre only once for our polytope Y and that this computation time is of the order of seconds.

7 Experiments

In line with past work on TSGs, we evaluate the performance of our approach on the airport passenger screening domain.

For the most simple head-to-head comparison, we look at the difference in solution quality between our approach and past work within single time-window. Brown et al. (2016) and McCarthy, Vayanos, and Tambe (2017) both have the same optimal solution in this case and the optimal marginal solution can be found by using a simple linear program (LP). When we compare the solutions of the LP to our approach, we control for the risk and measure the corresponding difference in delay. Specifically, we take the different risk levels associated with the uncontrollable categories ψ_θ from the solution of the LP and run our approach using those as the risk threshold in the risk constraints of our approach. We then test both sets of policies (LP and ours) using an online simulator and compare the ratio of average delays obtained.

We construct our problem instances using the description in Brown et al. (2016) and McCarthy, Vayanos, and Tambe (2017). The attacker utility associated with successfully launching an attack U^+ is sampled from a uniform distribution over [1, 10], while the utility of failing to launch an attack U^- is set to 0. The game is zero-sum and, as a result, the defender utilities are the negation of the attacker utilities. There are 3 attack methods m, 5 uncontrollable screenee risk levels θ and 5 screening resource types $|R|$. The efficacies (probability of detection) of different resources are sampled from a uniform random distribution over [0, 1] for each attack method. We create 10 random 2-sized combinations of resources to represent the teams T and their efficacies are found assuming that the efficacies of their associated resources are independent. We choose a passenger arrival distribution as used in McCarthy, Vayanos, and Tambe (2017), and consider arrivals to be normally distributed in a 3-hour window leading up to the departure of the flight. We combine this with real flight departure times taken from one of the busiest airports in the world to generate a realistic arrival distribution of passengers. The default number of flight types for experiments is 10.

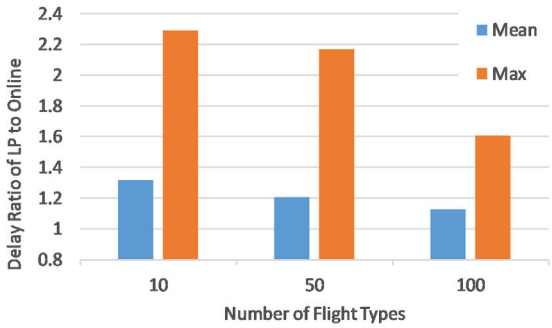

Figure 3: Static vs. Adaptive: We measure the ratio of average delays of the LP to average delays of our online approach (so higher is better) across different problem sizes.

Finally, for runtime, all past methods have used non-gradient based optimization methods and have reported runtimes for programs that have run on CPUs. As has been observed in other domains, GPUs offer a huge advantage due to the immense parallelization of matrix operations for neural network optimization. However, to perform a fair comparison to past work, we run all our experiments on a CPU. Thus, while our scalability results show the runtime trend with increasing problem size, the absolute wall clock time can be much better with GPUs.

Static vs. Adaptive

First, we show that our online approach outperforms past window-based approaches even when the problem size is large. The experiments here are evaluated for 30 random game instances and averaged across 100 samples of passenger arrival sequence. The results can be seen in Figure 3.

For the one time-window problem, improvement in solution quality comes from the fact that past work has a static policy within one time-window, whereas our solution can adapt based on the actual number of passenger arrivals. As a result, our approach can exploit the structure present within a time-window. The reason our improvement decreases with an increase in the number of flights is because, as the problem size increases, the structure present in a randomly generated problem decreases. For example, with many overlapping Gaussian distribution of passenger arrivals, the overall arrival is almost uniform which we show leads to a lower gain (see Section 7).

Moreover, the performance improvement within one

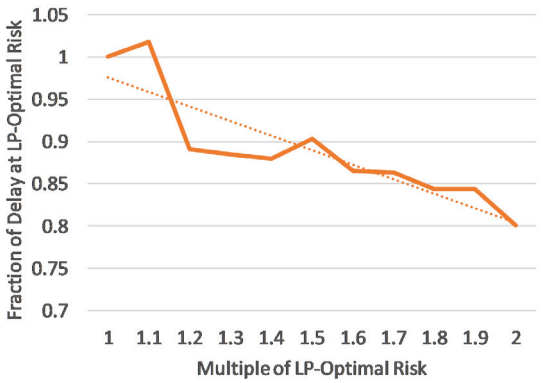

Figure 4: Delay vs. Risk: We measure how the delay decreases (as fraction of delay at LP optimal risk) with increasing risk allowance.

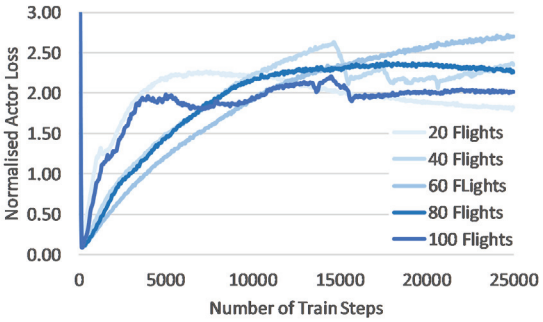

Figure 5: Scalability in training steps: We measure the actor loss of our DDPG network as a function of the number of mini-batches that are fed to it as input.

time-window is a lower bound on the amount of improvement we can achieve over past methods. The solution quality associated with past methods deteriorates with an increase in the number of windows (details in Section 2) while there is no notion of time-windows in our model and hence no degradation over long periods.

Delay vs. Risk

In Figure 4, we show the inherent trade-off between risk and average delay by varying ψ (keeping the ratio of ψ_θ fixed across θ) and measuring its effect on the delay. To do this, we take the optimal risk level obtained by solving the LP and then measure the impact on average delay as we relax it. The results show average delay as a fraction of the delay obtained at the LP optimal risk. The resulting curve can be seen as the Pareto frontier described in Section 4 restricted to the case where the ratio of ψ_θ fixed across θ. As expected, less stringent risk requirements result in lower average delay.

Scalability

In Figure 5, we show how training time is affected by the size of the game instance. Given that neural networks are not guaranteed optimal, measuring this is slightly challenging but we use the metric of the time at which our DDPG's actor network converges to measure how long the training time takes. As we see in the figure, the number of steps to convergence is about the same, regardless of the number of flights. Based on Figure 5, we choose 10,000 training steps as the number of steps for convergence.

Of course, as the input size increases with increasing number of flights, the time taken per training step increases, thus, the actual wall clock time to get to 10,000 steps for different number of flights varies. Figure 6 show the actual wall clock time for convergence (to 10,000 steps) with varying number of flights. The increase appears linear showing the scalability of our approach (as a reminder these results are not even using GPUs). In contrast, past work (Brown et al. 2016; McCarthy, Vayanos, and Tambe 2017) scale highly non-linearly with number of flights and have shown solutions only up to 50 flight and 15 flights respectively.

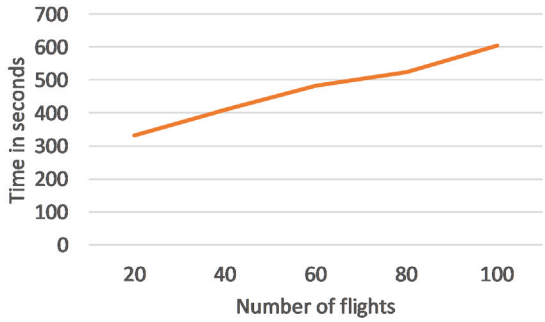

Figure 6: Scalability in actual time: We measure the time in secs at 10,000 training steps for different number of flights.

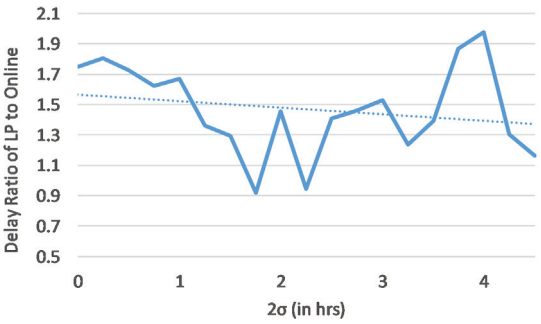

Figure 7: Delay vs. Variance: We measure the ratio of delays for policies computed by LP to our approach across different standard deviation for our arrival distribution.

Delay vs. Variance

In Figure 7, we look at how the variance associated with the passenger arrival distribution affects our gain over the time-window based solutions. Here the x-axis is measured using 2σ because of the intuition that 95% of passengers of a flight arrive within a 2 standard deviation window around

the mean. This graph can be interpreted as the effect that changing the width of the arrival window (of 95% passengers) has on solution quality. We vary it from 0 to 5 hours.

We find that as the variance associated with arrivals increases, the gain obtained by using an online approach as ours decreases. We believe that this is because the amount of structure present in the problem decreases as the variance increases. In the limit, when the variance is infinity, the arrivals are uniformly distributed, memory-less, and resemble a Poisson process. As a result, there is no information to be gained whenever the next passenger arrives, hence the per-passenger adaptive solution would do as good as one per time-window adaptive solution in this limiting case. Conversely, if the same number of flights arrive over a longer duration, say 24 hours, our algorithm would do considerably better in terms of the average delay since the arrival windows are less likely to overlap, resulting in smaller overall variance.

8 Conclusion

In summary, we proposed a novel model for threat screening that captures inherent features of the problem such as continuous arrival of screenees. We then provided an RL-based method to solve the model which includes the novel α-projection method for imposing hard constraints on actions. We believe these advances make our approach for threat screening realistic and applicable in practice. More broadly, our approach can be applied to solve zero-sum security games with sequential moves.

9 Acknowledgement

This research was supported by the Singapore Ministry of Education Academic Research Fund (AcRF) Tier 2 grant MOE2016-T2-1-174, Ministry of Education Academic Research Fund (AcRF) Tier 1 grant 19-C220-SMU-011 and the Lee Kong Chian Fellowship awarded by Singapore Management University. This work was also sponsored by the Army Research Office and accomplished under MURI Grant Number W911NF-17-1-0370. Perrault was supported by the Center for Research on Computation and Society.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; and Zheng, X. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 22–31. JMLR. org.

Amos, B., and Kolter, J. Z. 2017. Optnet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 136–145. JMLR. org.

Balcan, M.-F.; Blum, A.; Haghtalab, N.; and Procaccia, A. D. 2015. Commitment without regrets: Online learning in stackelberg security games. In *Proceedings of the sixteenth ACM EC*, 61–78. ACM.

Basilico, N.; Gatti, N.; and Amigoni, F. 2009. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, 57–64. International Foundation for Autonomous Agents and Multiagent Systems.

Bhatia, A.; Varakantham, P.; and Kumar, A. 2019. Resource constrained deep reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, 610–620.

Bosansky, B.; Jiang, A. X.; Tambe, M.; and Kiekintveld, C. 2015. Combining compact representation and incremental generation in large games with sequential strategies. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Brown, M.; Sinha, A.; Schlenker, A.; and Tambe, M. 2016. One size does not fit all: A game-theoretic approach for dynamically and effectively screening for threats. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Černỳ, J.; Boỳanskỳ, B.; and Kiekintveld, C. 2018. Incremental strategy generation for stackelberg equilibria in extensive-form games. In *Proceedings of the 2018 ACM EC*, 151–168. ACM.

Chow, Y.; Ghavamzadeh, M.; Janson, L.; and Pavone, M. 2017. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research* 18(1):6070–6120.

Dastin, J. 2016. Thousands miss flights because of airport screening: American airlines executive. https://www.reuters.com/article/us-usa-security-american-airline/thousands-miss-flights-because-of-airport-screening-american-airlines-executive-idUSKCN0YH1KV. Online; accessed 2 Sep 2019.

Delle Fave, F. M.; Jiang, A. X.; Yin, Z.; Zhang, C.; Tambe, M.; Kraus, S.; and Sullivan, J. P. 2014. Game-theoretic patrolling with dynamic execution uncertainty and a case study on a real transit system. *Journal of Artificial Intelligence Research* 50:321–367.

Kamra, N.; Gupta, U.; Fang, F.; Liu, Y.; and Tambe, M. 2018. Policy learning for continuous space security games using neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Kamra, N.; Gupta, U.; Wang, K.; Fang, F.; Liu, Y.; and Tambe, M. 2019. Deep fictitious play for games with continuous action spaces. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19.

Kroer, C.; Farina, G.; and Sandholm, T. 2018. Robust stackelberg equilibria in extensive-form games and extension to

limited lookahead. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Letchford, J., and Conitzer, V. 2010. Computing optimal strategies to commit to in extensive-form games. In *Proceedings of the 11th ACM conference on Electronic commerce*, 83–92. ACM.

Letchford, J.; MacDermed, L.; Conitzer, V.; Parr, R.; and Isbell, C. L. 2012. Computing optimal strategies to commit to in stochastic games. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Letchford, J.; Conitzer, V.; and Munagala, K. 2009. Learning and approximating the optimal strategy to commit to. In *International Symposium on Algorithmic Game Theory*, 250–262. Springer.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Maxwell, M. S.; Restrepo, M.; Henderson, S. G.; and Topaloglu, H. 2010. Approximate dynamic programming for ambulance redeployment. *INFORMS Journal on Computing* 22(2):266–281.

McCarthy, S. M.; Laan, C. M.; Wang, K.; Vayanos, P.; Sinha, A.; and Tambe, M. 2018. The price of usability: Designing operationalizable strategies for security games. In *Proceedings of theTwenty-Seventh International Joint Conference on Artificial Intelligence: IJCAI-18*, 454–460.

McCarthy, S. M.; Vayanos, P.; and Tambe, M. 2017. Staying ahead of the game: Adaptive robust optimization for dynamic allocation of threat screening resources. In *IJCAI*, 3770–3776.

Pham, T.-H.; De Magistris, G.; and Tachibana, R. 2018. Optlayer-practical constrained optimization for deep reinforcement learning in the real world. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 6236–6243. IEEE.

Schlenker, A.; Xu, H.; Guirguis, M.; Kiekintveld, C.; Sinha, A.; Tambe, M.; Sonya, S. Y.; Balderas, D.; and Dunstatter, N. 2017. Don't bury your head in warnings: A game-theoretic approach for intelligent allocation of cybersecurity alerts. In *IJCAI*.

Simao, H. P.; Day, J.; George, A. P.; Gifford, T.; Nienow, J.; and Powell, W. B. 2009. An approximate dynamic programming algorithm for large-scale fleet management: A case application. *Transportation Science* 43(2):178–197.

Wang, Y.; Shi, Z. R.; Yu, L.; Wu, Y.; Singh, R.; Joppa, L.; and Fang, F. 2019. Deep reinforcement learning for green security games with real-time information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 1401–1408.

Wright, M.; Wang, Y.; and Wellman, M. P. 2019. Iterated deep reinforcement learning in games: History-aware training for improved stability. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, 617–636. ACM.