Adversarial Reinforcement Learning

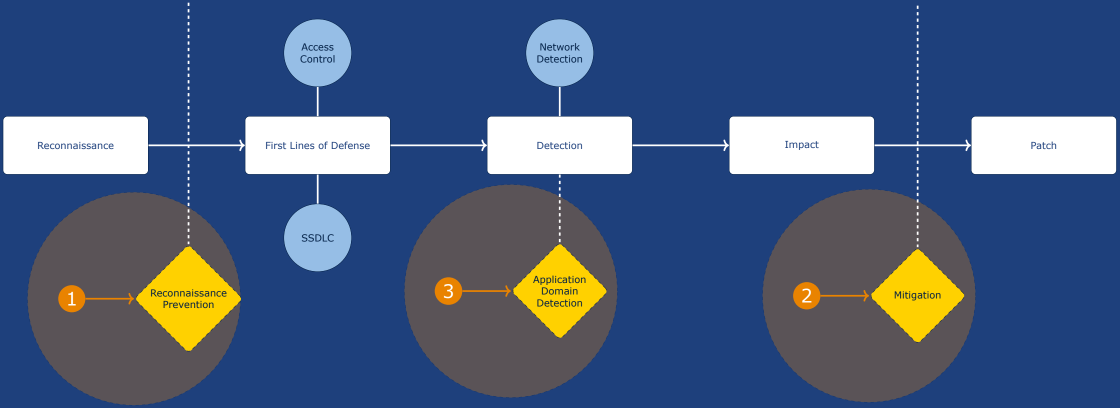
for Cyber-Attack Prevention, Detection, and Mitigation

Taha Eghtesad

Comprehensive Exam
March 15th, 2024

PennState
College of Information Sciences and Technology

Cyber-Attack Life-Cycle



PennState
College of Information Sciences and Technology

Moving Target Defense I

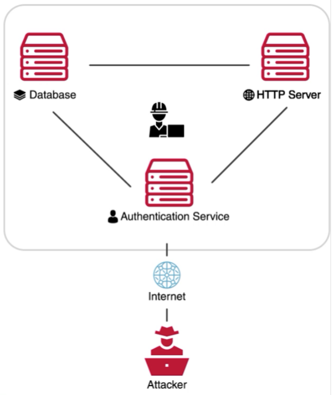
Preventing reconnaissance using Moving Target Defense (MTD)

MTD is a *proactive* defense

Changing the configuration of assets randomly.
e.g., IP addresses, software deployments

Increases the uncertainty of the attacks.

Putting the adversary in an infinite loop of exploration



PennState
College of Information Sciences and Technology

Moving Target Defense II

MTD configurations should be deployed continuously.

Currently, sysadmins manually decide on **when** and **where** MTD configurations to be deployed based on their **experience**.

Deployment is time-consuming

- Constraint on deployment locations.
- Physical connectivity cannot be changed.
- Resources are limited

The trade-off between security and efficiency

- **Most Secure**: Total Randomization of configurations
- **Most Efficient**: No change of configuration.

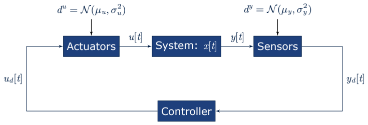
Mitigation of *0-stealthy* Attacks

0-stealthy against **Physical Control Software**

- Insider threats are **undetectable** by network intrusion systems
- Change the actuation and observation signal
- Change is in the operational domain
- Make the system deviate from its nominal operations. *e.g.*, StuxNet Virus

- *These systems can not be easily patched.*
- *These systems can not be easily restarted.*

Mitigating the effect of worst-case attack by adjusting the actuator signals.



From Associated Press

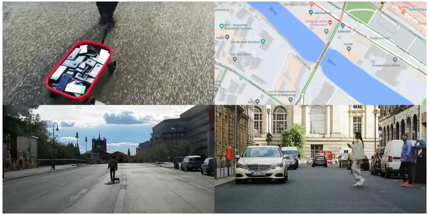
Domain Specific Detection

Relying on network and host intrusion detection is not enough.

Misinformation in Transportation Networks

- *Changing Road Signs*
- *False Data Injection in Crowdsourced Information (Google Maps)*
- *Manipulate Traffic Signals*

- **Detection** must happen in the **application domain**.
- Must detect **system deviations** from its nominal operating conditions.
- Taking operational requirements into account.



From [Schoon, 2020]

PennState College of Information Sciences and Technology

Reinforcement Learning for Decision Making Under Uncertainty

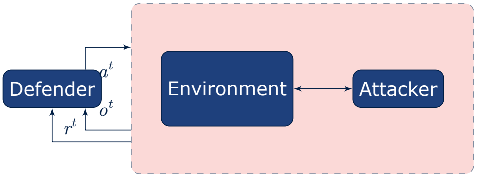
Reinforcement Learning for sequential decision making under uncertainty

- In *prevention*, RL can determine timing of **MTD configurations** considering uncertain threat scenarios.
- In *detection*, RL can make decisions to **maximize detection accuracy** under uncertain system conditions.
- In *mitigation*, RL can be employed to **minimize potential damage** by selecting appropriate actuation signals in uncertain environments.

Reinforcement Learning I

Reinforcement Learning (RL) is a Sequential Decision-Making Algorithm **under uncertainty**

- Relies on trial-and-error and dynamic programming to achieve **Optimal Sequential Decisions**
- Learns from *experiences* to maximize expected **Reward**



Reinforcement Learning II

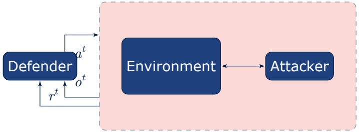
Reinforcement Learning Optimization Problem

Policy or **Strategy** Function

$$\boxed{\pi}(o^t) \mapsto a^t$$

Optimization Problem

$$\max \mathbb{E}\left[\sum_{\tau=0}^{\infty} \gamma^{\tau} \cdot r^{t+\tau} \;\middle|\; \pi\right]$$



In English Please?

*The **policy** function gets the current observation and suggests an action that maximizes "discounted future rewards"*

Discount Factor γ

$\gamma \in [0, 1)$ *prioritizes rewards received in the current time step over future rewards.*

- $\gamma = 0$: *Only care about the current reward*
- $\gamma = 1$: *All future rewards are equal*

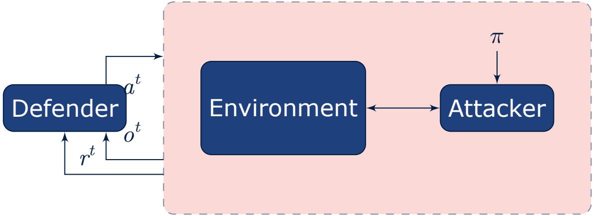
PennState College of Information Sciences and Technology

Research Questions

RQ 1 How to model the interactions between the **attacker** and **defender** within the application **environment**?

RQ 2 What Reinforcement Learning **algorithm** should we use or develop?

RQ 3 How can we learn if the attacker is **adaptive** and responds to the defender?



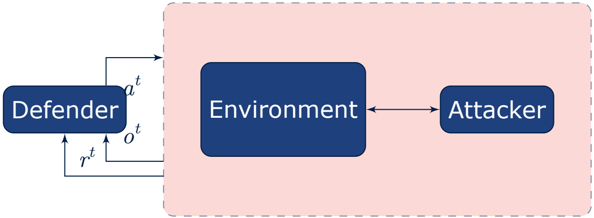
Research Approach

PennState
College of Information
Sciences and Technology

The Interactions

For each of the **Prevention**, **Detection**, and **Mitigation** scenarios, we need to formalize the interactions between the attacker, environment, and the defender.

- What is the **environment model**?
- What is the **threat model**?
- What is the **defender model**?



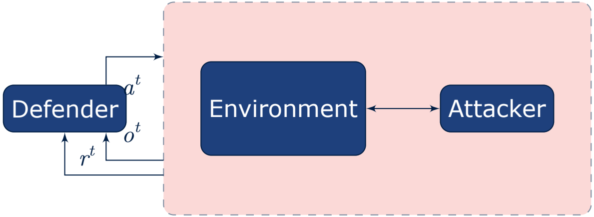
Reinforcement Learning Algorithm

Deep-Q-Learning

- **Deterministic** Policy
- **Discrete** Action
- Continuous or Discrete Observation

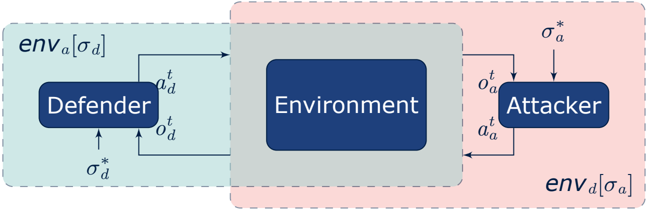
Deep Deterministic Policy Gradients

- **Deterministic** Policy
- **Continuous** Action
- Continuous or Discrete Observation



But, what if the attacker is adaptive?

- An **adaptive** attacker is not static. It has its own **policy** and adapts it to the defender
- The defender must also adapt to the attacker



What is that σ?

That is a stochastic policy, i.e., a probability distribution over deterministic policies $\boldsymbol{\pi}$.

Two Player Security Game

- The *interactions* and *objectives* of players is expressed as a **Multi-Agent Partially Observable Markov Decision Process (MAPOMDP)**.
- MAPOMDP is a relaxed **Extensive-Form Game**.

MAPOMDP

$$\langle P, \mathcal{S}, \{\mathcal{A}_p\}_{p \in P}, \mathcal{T}, \{\mathcal{R}_p\}_{p \in P}, \{\mathcal{O}_p\}_{p \in P} \rangle$$

What are these symbols?

P set of **players**

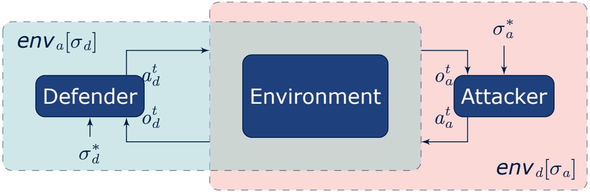
$\mathcal{S}$ set of **states**

$\mathcal{A}_p$ **action space** of player p

$\mathcal{T}(s, \mathbf{a})$ **state transitions** rules

$\mathcal{R}_p(s, \mathbf{a})$ **rewarding** rule for player p

$\mathcal{O}_p$ **observation** rule for player p



Finding Equilibrium of a Security Game

- We defined a two-player game between the attacker and the defender as a MAPOMDP.
- We assume both players are rational. They always choose a **best-response strategy**.
- This is equivalent to finding the Nash Equilibrium of the security game.

What is Mixed-Strategy Nash Equilibrium?

All players are playing with their best-response to all opponents' strategies, i.e., neither player can increase their expected utility without having their opponents change their strategy.

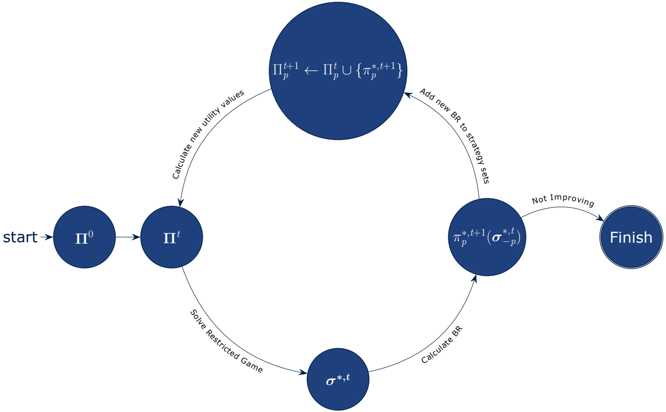
Problem

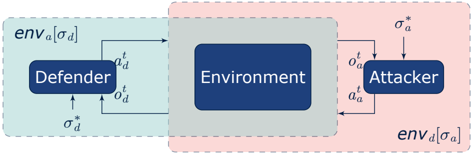
Enumerating all the different strategies to find the Nash Equilibrium is **infeasible**.

Reinforcement Learning as Best-Response Oracle

Reinforcement Learning algorithms can be used to find an *approximate* Best-Response.

Policy Space Response Oracles

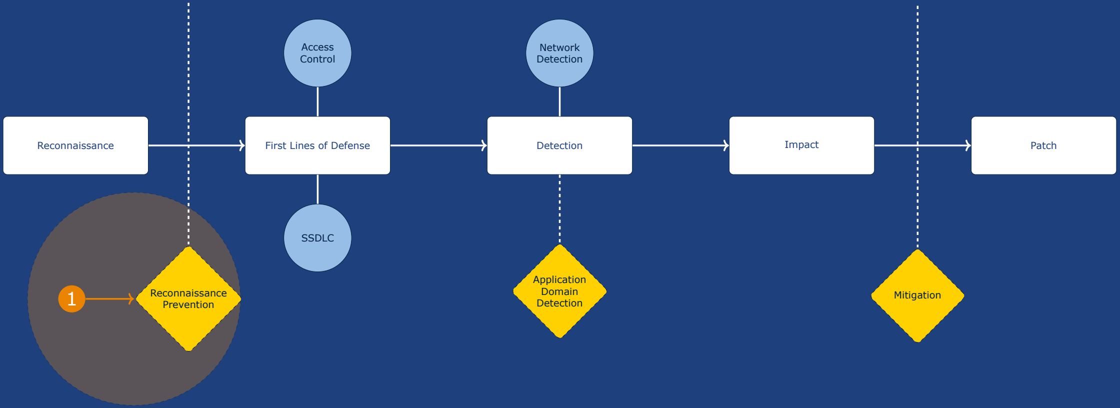




PSRO algorithm [Lanctot et al., 2017] based on Double Oracles [McMahan et al., 2003].

PennState College of Information Sciences and Technology

Attack Prevention through Moving Target Defense [Eghtesad et al., 2020]



PennState College of Information Sciences and Technology

The Moving Target Defense Game I

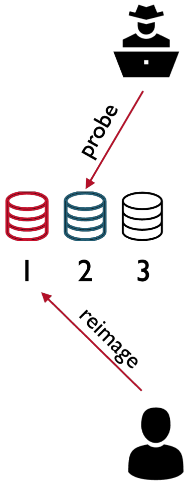
Overview

The Game [Prakash and Wellman, 2015]

An Attacker and a Defender compete over a set of servers.

- Adversary **probes** a server
- Adversary compromises the server with some probability, or
- Increases the chance of compromising that server in the future.

- Defender **reimages** a server.
- Takes the server down for fixed time steps.
- Resets the adversary's progress on that server
- Takes back the control of the server



The Moving Target Defense Game II

States, Objectives, and Rewards

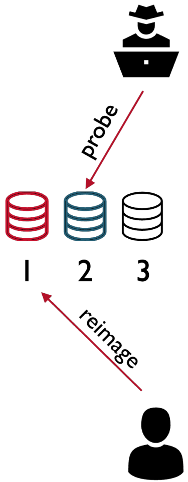
State per each Server

- Number of Probes
- Control
- Up or Time to up

Rewards

Each player is rewarded based on the portion of servers that are in control or down.

- Implicit defender cost, *i.e.*, **not gaining reward** when server is down
- Explicit attacker reward penalty for probing.



The Moving Target Defense Game III

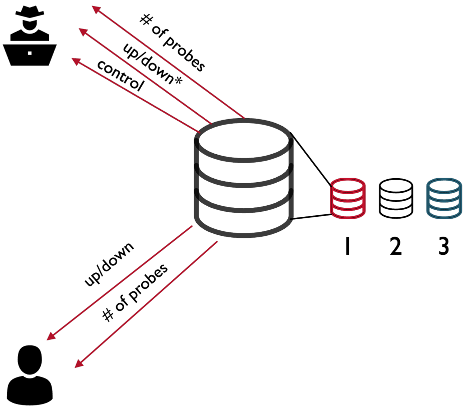
Observations

Attacker Knowledge

- Can only estimate when a server is compromised.
- Learns whether a server is <u>up or down</u> by probing.
- Knows who <u>controls</u> a server.
- Knows when a compromised server is reimaged.

Defender Knowledge

- Knows Which servers are down.
- Observes a probe with some probability.
- Unaware of a server being compromised.



Imperfect Observation

The state is only partially observable

The defender have observed only a few probes recently

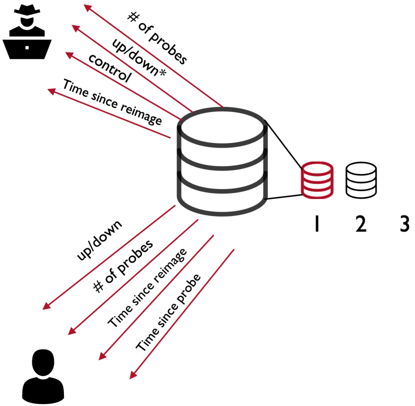
- *Is the server already compromised? The attacker is not probing it*
- *Is the server not compromised? There are few probes*

The attacker is unaware of an unprobed server

- *Has its progress been reset by reimaging?*
- *The previous probes are still in place?*

Including History

Improve RL with <u>compact history representation</u>



Short-term Losses vs. Long-term Rewards

Short Term Loss

- **Defender**: Reimaging a server results in lower rewards while the server is offline
- **Attacker**: Probing incurs a cost

Long-term Gains

- **Defender**: A reimaged server will return the control to the defender
- **Attacker**: Continuous probes will compromise a server

The value of *discount factor* γ is important

Deep-Q-Learning for MTD

What Reinforcement Learning Algorithm to Choose?

- At each step, both agents decide on **a server** to probe or reimage
- Action space is **discrete**
- Observation features are **well-defined**
- *Feed Forward Neural Network* operates as a feature extractor
- **Deep-Q-Learning algorithm** can be used as the best-response oracle for both agents
- *Policy Space Response Oracles* will find the **Nash Equilibrium** of the MTD game

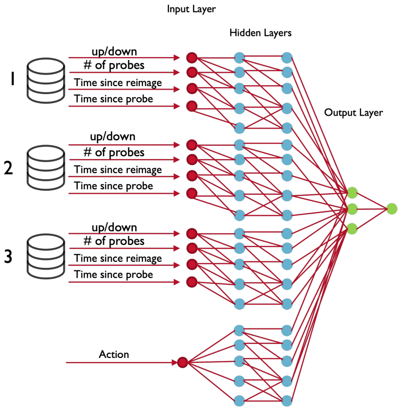


Figure 1: Defender NN Architecture

Evaluation

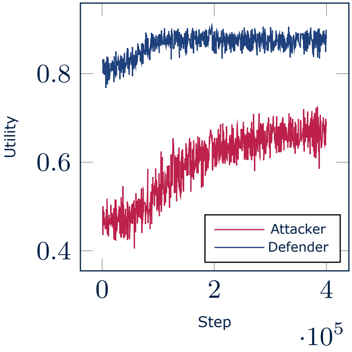


Figure 2: RL Training Curve for the first PSRO iteration

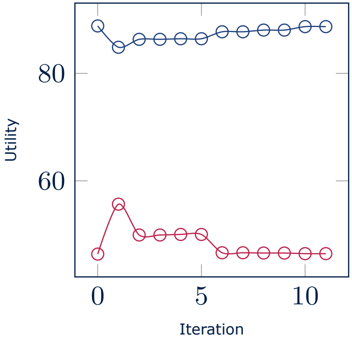


Figure 3: PSRO curve value after each iteration

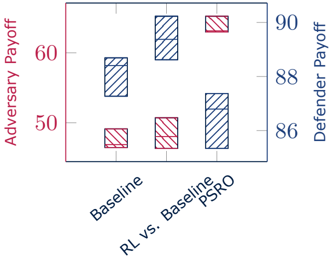
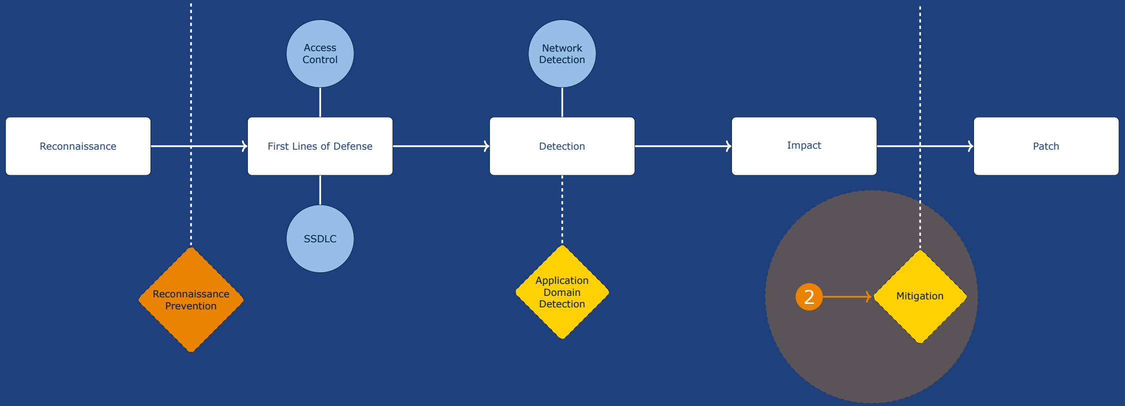


Figure 4: Episodic reward

Mitigation of False Data Injection in Industrial Control Systems



PennState
College of Information Sciences and Technology

Control System In Danger I

System Transitions

Differential equations dictate the transition rules

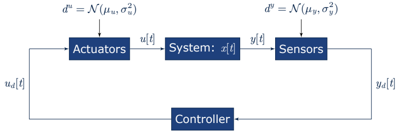
Attacker has Compromised Signals

- Will change sensor or actuator signals by a percentage to avoid detection

Presence of a Detector

A detector notifies the controller

[Giraldo et al., 2019, Paridari et al., 2018, Urbina et al., 2016]



From Associated Press

Control System In Danger II

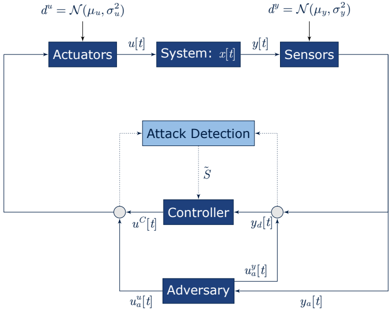
Observations, Actions, and Objectives

Attacker

- **Observes** sensor and actuator values
- **Perturbs** sensor and actuator values
- **Deviates** the system from nominal point

Defender

- **Observes** perturbed sensor signals
- **Observes** detection result
- **Decides** on control signals
- Minimize **worst-cast** deviations of nominal point



- **Attacker** gains reward by distance of system state to its nominal point $r_a = \|x - x_0\|$
- **Defender** gains reward by the negative of the distance $r_d = -\|x - x_0\|$

Resilient Control through Reinforcement Learning

What Reinforcement Learning Algorithm to Choose?

- Action space is **continuous**
- Observation features are **well-defined**
- *Feed Forward Neural Network* operates as a feature extractor
- **Deep Deterministic Policy Gradients (DDPG)** can be used as the best-response oracle for both agents
- *Policy Space Response Oracles* will find the **Nash Equilibrium**, thus the resilient control policy

Evaluation I

Benchmark Systems

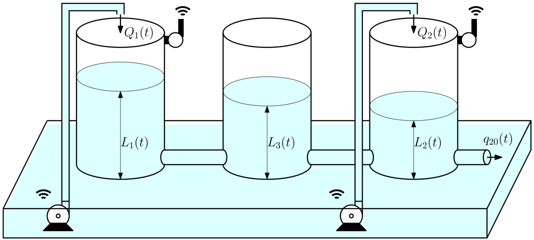


Figure 5: Schema of the **three tanks** system
Diagram from [Combita et al., 2019]

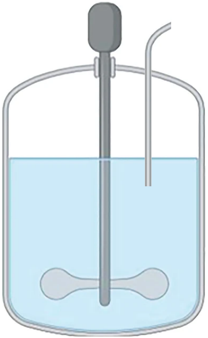


Figure 6: Schema of a **bioreactor**
Diagram from [Rahmatnejad et al., 2023]

Evaluations II

Bioreactor

Observation attacks cannot be tackled. There is no correlation between observation and reward.

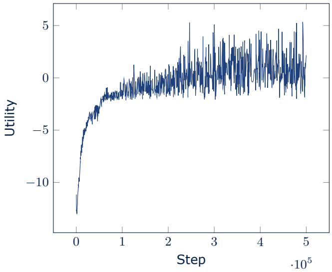


Figure 7: Learning curve of first iteration of DDPG in bioreactor

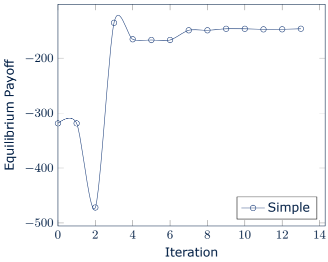


Figure 8: Episodic rewards in iterations of PSRO.

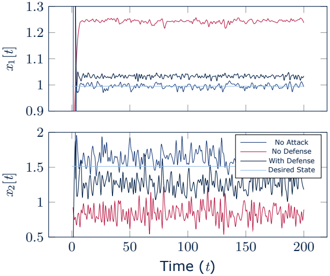


Figure 9: State Comparison

Evaluations II

Three Tanks

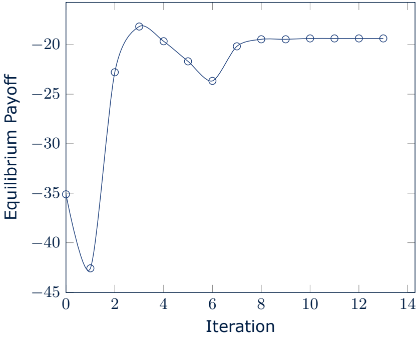


Figure 10: Episodic rewards in iterations of PSRO.

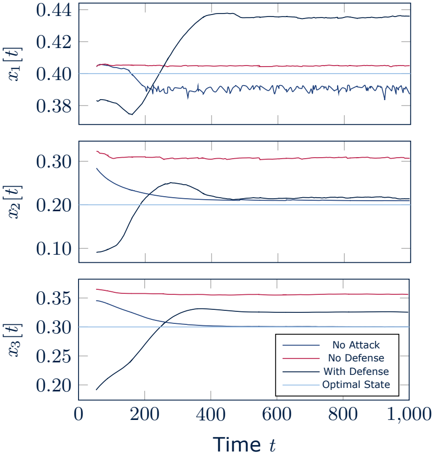
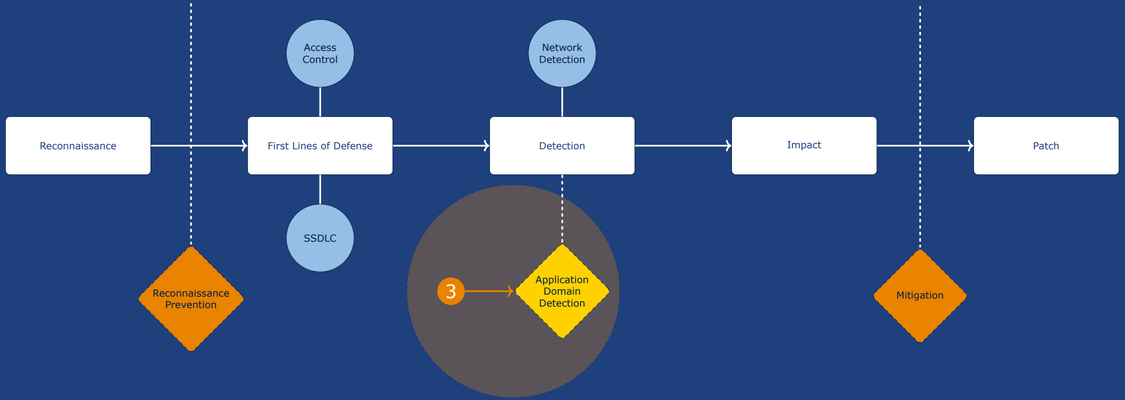


Figure 11: State Comparison

Detection of False Data Injection in Navigation Applications [Eghtesad et al., 2023]



PennState
College of Information Sciences and Technology

Attack on Navigation Applications I

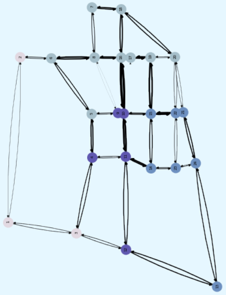
Transportation Model

- A bi-directional **graph** defines the transportation network's roads and intersections
  [Transportation Networks for Research Core Team, 2020]
- Each road has a given travel time: **Congestion Model**
- The more vehicles on a given road, the higher the travel time
- At each intersection, drivers look at the shortest path to destination by **the navigation application**

Threat Model

- Attacker has a budget to perturb perceived travel times
- Attacker perturbs perceived travel time
- Drivers take a longer path due to **perceived congestion**

Sioux Falls, ND



Attack on Navigation Applications II

Defense Model

- **Detection** of suspicious activity
- Detector raises an alarm
- Attack is defused if correctly detected

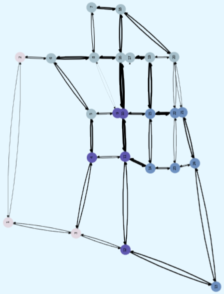
Observations

- Attacker has full observation of the network: vehicles, vehicle locations, driver destinations
- Detector only observes **reported** perceived travel time

Objectives

- **Attacker** gains reward by maximizing the total travel time
- **Detector** prevents increase in total travel time

Sioux Falls, ND



PennState College of Information Sciences and Technology

Attack detection through Reinforcement Learning

Let's try PSRO

- The attacker has full graph observation
- The attacker has a constrained continuous action
- The detector has a partial graph observation
- The detector takes a boolean action, either to raise an alarm or not

DDPG as attack oracle

- The attacker should output perturbations for thousands of city roads
- General-Purpose Reinforcement Learning algorithms are infeasible even for a small city
- It requires millions of samples collected from the environment
- We need a robust and **feasible** attack oracle

Hierarchical Multi-Agent Reinforcement Learning as Attack Oracle

Idea

- We can divide the network into **components** of smaller size.
- **Low-Level** RL agents are assigned to each component
- A **High-Level** RL agent coordinates the low-level agents

Why a high-level coordinator?

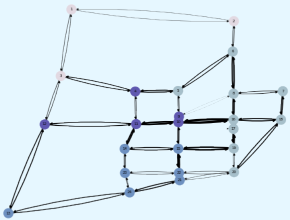
- *The total perturbations are restricted by a budget*
- *Component agents compete over the budget*

- The high-level agent allocates the perturbation budget to the component agents
- The low-level agents distribute given perturbation allocation to road links

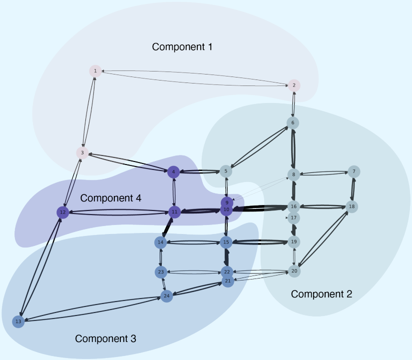
Network Decomposition

Decompose network based on **K-Means Clustering** by edge distance without congestion

Original Sioux Falls Network



Decomposed Sioux Falls Network



PennState College of Information Sciences and Technology

Hierarchical Approach I

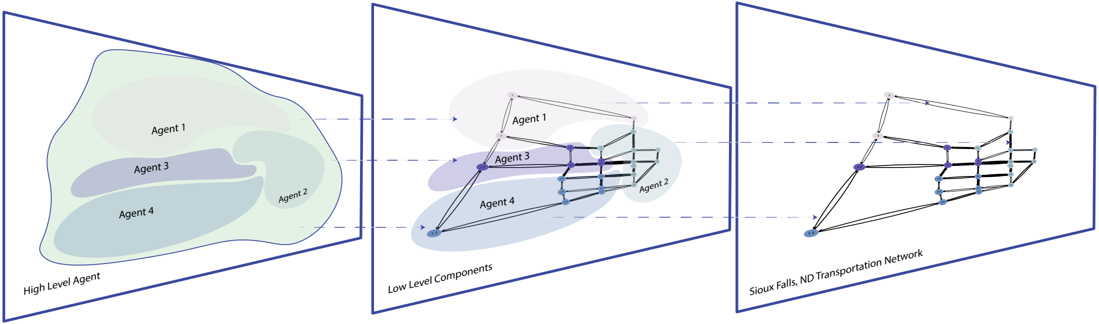
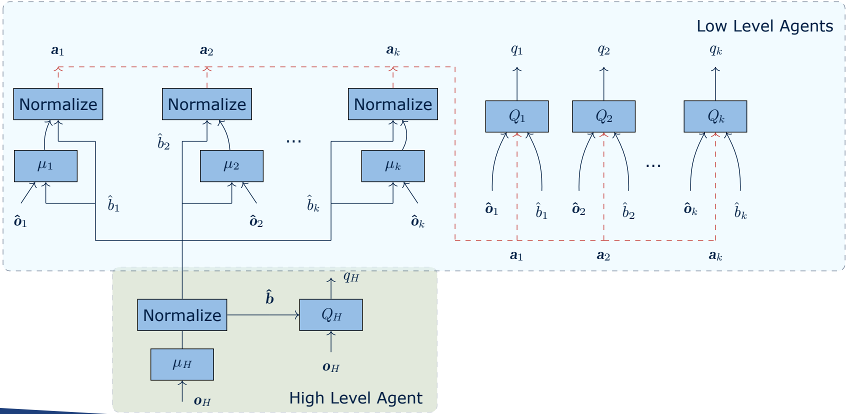


Figure 12: Hierarchical Multi-Agent Reinforcement Learning

Hierarchical Approach II

Reinforcement Learning Model



Hierarchical Approach III

Reinforcement Learning Observations

Each Low-Level observes 5 features per each edge e

1. Number of vehicles that are at an intersection with an unperturbed shortest path to the destination that passes through e
2. Number of vehicles currently on e
3. Number of vehicles that are at an intersection that will immediately take e as their shortest path without perturbation
4. Sum of remaining travel times of vehicles currently on edge e
5. Number of vehicles that are on an edge but will take e as the shortest path

High-Level observes

The sum of each feature per component as **summary**

PennState College of Information Sciences and Technology

Hierarchical Approach IV

Rewards and Updating

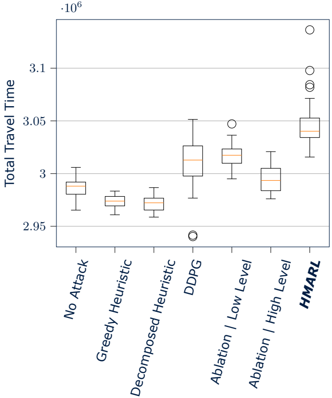
Aim of the attacker is to maximize total travel time

- Low-level agents gain reward by the **number of vehicles in its component**
- High-level agent gain reward by the **total number of vehicles**
- This is equivalent to maximizing the total travel time.

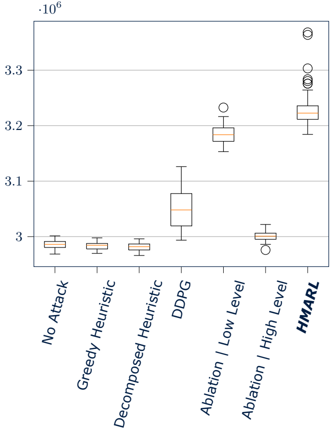
Since the agents cooperate, all agents can be trained simultaneously.

Evaluation

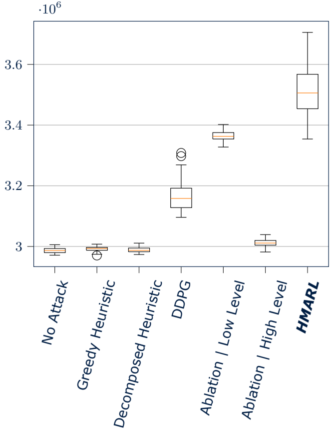
Budget $B = 5$



Budget $B = 10$



Budget $B = 15$



Future Plans and Timeline

PennState
College of Information
Sciences and Technology

Published Work

- **T. Eghtesad** et al; Adversarial Deep Reinforcement Learning based Adaptive Moving Target Defense; **Conference on Decision and Game Theory for Security (GameSec'20)**
- **T. Eghtesad** et al; Hierarchical Multi-Agent Reinforcement Learning for Assessing False-Data Injection Attacks on Trans-portation Networks; **International Conference on Autonomous Agents and Multiagent Systems (AAMAS'24)**; *accepted for publication*

Future Work and Timeline

- **Spring 2024**: Finalize the threat and defense model of false data injection detection in navigation applications.
- **Spring 2024**: Adjust our HMARL framework for the new threat model.
- **Summer 2024**: Development of the PSRO algorithm for detection of false data injection in transportation networks assuming a *black-box* model where neither the adversary nor the defender has access to the opponent's strategies and only observes its consequences.
- **Fall 2024**: Extend and develop the mitigation model with larger, more realistic ICS for the mitigation tasks.

Other Published Articles

- O. Akgul, **T. Eghtesad**, et al; Bug Hunters' Perspectives on the Challenges and Benefits of the Bug Bounty Ecosystem; *USENIX Security'23*; **Core Ranking A***; **Distinguished Paper Award (Top %5)**
- O. Akgul, **T. Eghtesad**, et al; Exploring Challenges and Benefits of Bug-Bounty Programs; WSIW'20
- S. Eisele, **T. Eghtesad**, et al; Safe and Private Forward-Trading Platform for Transactive Microgrids; TCPS; Dec 2020
- S. Eisele, **T. Eghtesad**, et al; Blockchains for Transactive Energy Systems: Opportunities, Challenges, and Approaches; IEEE Computer; Sep 2020
- C. Barreto, **T. Eghtesad**, et al; Cyber-attacks and mitigation in blockchain based transactive energy systems; ICPS'20
- S. Eisele, **T. Eghtesad**, et al; Decentralized Computation Market for Stream Processing Applications; IC2E'22
- S. Eisele, **T. Eghtesad**, et al; Mechanisms for Outsourcing Computation via a Decentralized Market; DEBS'20

References I

[Combita et al., 2019] Combita, L. F., Cardenas, A., and Quijano, N. (2019).
Mitigating Sensor Attacks Against Industrial Control Systems.
IEEE Access, 7:92444–92455.

[Eghtesad et al., 2023] Eghtesad, T., Li, S., Vorobeychik, Y., and Laszka, A. (2023).
Hierarchical Multi-Agent Reinforcement Learning for Assessing False-Data Injection Attacks on Transportation Networks.

[Eghtesad et al., 2020] Eghtesad, T., Vorobeychik, Y., and Laszka, A. (2020).
Adversarial Deep Reinforcement Learning Based Adaptive Moving Target Defense.
In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12513 LNCS, pages 58–79.

[Giraldo et al., 2019] Giraldo, J., Urbina, D., Cardenas, A., Valente, J., Faisal, M., Ruths, J., Tippenhauer, N. O., Sandberg, H., and Candell, R. (2019).
A Survey of Physics-Based Attack Detection in Cyber-Physical Systems.
ACM Computing Surveys, 51(4):1–36.

[Lanctot et al., 2017] Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Pérolat, J., Silver, D., and Graepel, T. (2017).
A unified game-theoretic approach to multiagent reinforcement learning.
Advances in neural information processing systems, 30.

[Lillicrap et al., 2015] 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.

[McMahan et al., 2003] McMahan, H. B., Gordon, G. J., and Blum, A. (2003).
Planning in the presence of cost functions controlled by an adversary.
In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 536–543.

References II

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015).
Human-level control through deep reinforcement learning.
Nature, 518(7540):529–533.

[Paridari et al., 2018] Paridari, K., O'Mahony, N., El-Din Mady, A., Chabukswar, R., Boubekeur, M., and Sandberg, H. (2018).
A Framework for Attack-Resilient Industrial Control Systems: Attack Detection and Controller Reconfiguration.
Proceedings of the IEEE, 106(1):113–128.

[Prakash and Wellman, 2015] Prakash, A. and Wellman, M. P. (2015).
Empirical Game-Theoretic Analysis for Moving Target Defense.
In *Proceedings of the Second ACM Workshop on Moving Target Defense*, pages 57–65, New York, NY, USA. ACM.

[Rahmatnejad et al., 2023] Rahmatnejad, V., Wei, Y., and Rao, G. (2023).
Recent Developments in Bioprocess Monitoring Systems.
pages 39–66.

[Schoon, 2020] Schoon, B. (2020).
Google Maps 'hack' uses 99 smartphones to create virtual traffic jams.

[Schulman et al., 2015] Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. (2015).
Trust Region Policy Optimization.

[Schulman et al., 2016] Schulman, J., Moritz, P., Levine, S., Jordan, M. I., and Abbeel, P. (2016).
High-Dimensional Continuous Control Using Generalized Advantage Estimation.

References III

[Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017).
Proximal Policy Optimization Algorithms.

[Transportation Networks for Research Core Team, 2020] Transportation Networks for Research Core Team (2020).
Transportation Networks for Research.
https://github.com/bstabler/TransportationNetworks/.

[Urbina et al., 2016] Urbina, D. I., Giraldo, J. A., Cardenas, A. A., Tippenhauer, N. O., Valente, J., Faisal, M., Ruths, J., Candell, R., and Sandberg, H. (2016).
Limiting the Impact of Stealthy Attacks on Industrial Control Systems.
In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1092–1105, New York, NY, USA. ACM.

Appendix

Nash Equilibrium

Policies and Strategies I

Pure Strategy

- A ***Strategy*** is a policy function that, given the current observation from the environment, produces an action to be taken by the agent.
- A ***Pure Strategy*** is a deterministic policy function.
- The ***Pure Strategy Set*** is the set of all possible pure strategies available to the player.
- A ***Pure Strategy Profile*** is a combination of all players strategy.
- The ***Utility Profile*** for player p is the amount of utility or reward the player p receives when players use a given strategy profile.

Pure Strategy

$$\pi(o) \mapsto a$$

Pure Strategy Set

$$\pi_p \in \Pi_p$$

Pure Strategy Profile

$$\boldsymbol{\pi} \leftarrow \{\pi_1, \pi_2, \cdots, \pi_p\}$$

Utility Profile

$$U_p(\boldsymbol{\pi})$$

Policies and Strategies II

Mixed Strategy

We need a mechanism to represent stochastic policies.

- A ***Mixed Strategy*** is a probability distribution over the player's pure strategy set.
- The ***Mixed Strategy Set*** is the set of all mixed strategies available to the player.
- The Mixed Strategy Utility Profile can be calculated using Pure Strategy Utility Profile.

Mixed Strategy Utility Profile

Utility from $\boldsymbol{\pi}$ times the probability that $\boldsymbol{\pi}$ occurs summed over all strategy profiles $\boldsymbol{\pi} \in \boldsymbol{\Pi}$.

Mixed Strategy

$$\boxed{\sigma_p(\pi_p)} \in [0, 1]$$

$$\sum_{\pi_p}^{\Pi_p} \sigma_p(\pi_p) = 1$$

Mixed Strategy Set

$$\sigma_p \in \Sigma_p$$

Mixed Strategy Profile

$$\boldsymbol{\sigma} \leftarrow \{\sigma_1, \sigma_2, \cdots, \sigma_P\}$$

Utility Profile

$$U_p(\boldsymbol{\sigma}) = \sum_{\boldsymbol{\pi} \in \boldsymbol{\Pi}} \left(\prod_p^P \sigma_p(\pi_p) \right) \cdot U_p(\boldsymbol{\pi})$$

Policies and Strategies II

Best Response and Nash Equilibrium

- All players are **rational**. Thus, they pick a strategy that maximizes their utility.
- A ***Best-Response*** Mixed Strategy provides maximum utility for the player given the strategy of opponents:
- Assuming all players are using their best response, the strategy profile is a ***Mixed Strategy Nash Equilibrium*** (MSNE).

Best-Response Mixed Strategy

$$\boxed{\sigma_p^*}(\boldsymbol{\sigma}_{-p}) = \underset{\sigma_p \in \Sigma_p}{\operatorname{argmax}}\, U_p(\{\sigma_p, \boldsymbol{\sigma}_{-p}\})$$

Mixed Strategy Nash Equilibrium

$$\forall_{p \in P} \forall_{\sigma_p \in \Sigma_p} : \; U_p(\boldsymbol{\sigma}^*) \geq U_p(\{\sigma_p, \boldsymbol{\sigma}^*_{-p}\})$$

What is Mixed-Strategy Nash Equilibrium?

All players are playing with their best-response to all opponents' strategies, and neither player can increase their expected utility without having their opponents change their strategy.

Independent Reinforcement Learning

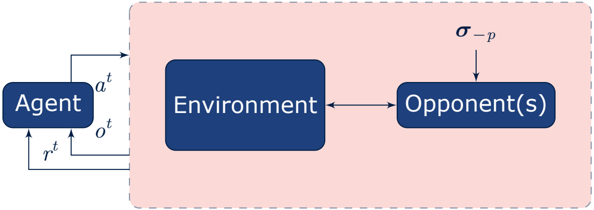
Reinforcement Learning Formalized I

Deep-Q-Learning I

- Extension of Q-Learning by [Mnih et al., 2015]
- Deterministic
- Off-Policy
- Value Iteration
- Action-Value Method
- Model-Free

RL Optimization Problem

$$\text{optimize } \pi(o^t) \mapsto a^t$$
$$\text{s.t. } \max \mathbb{E}\left[\sum_{\tau=0}^{\infty} \gamma^{\tau} \cdot r^{t+\tau} \,\middle|\, \pi, \boldsymbol{\sigma}_{-p}\right]$$



Reinforcement Learning Formalized II

Deep-Q-Learning II

The Parametric Q Function

$$Q^{\theta}(o^t, a^t) = r^t + \mathbb{E}\left[\sum_{\tau=1}^{\infty} \gamma^{\tau} \cdot r^{t+\tau} \;\middle|\; \pi\right]$$

Bellman Mean Squared Error

$$L(\theta) = \frac{1}{|X|} \sum_{x}^{X} \left(\boxed{r_x^t + \gamma \cdot \operatorname*{argmax}_{a'} Q^{\theta}(o_x^{t+1}, a')} - \boxed{Q^{\theta}(o_x^t, a_x^t)} \right)^2$$

Policy Function

$$a^t \leftarrow \pi(o^t) = \max_{a'} Q(o^t, a')$$

Experience

$$x = \langle o_x^t, a_x^t, o_x^{t+1}, r_x^t \rangle \in X \sim E$$

PennState College of Information Sciences and Technology

Reinforcement Learning Formalized III

Deep Deterministic Policy Gradients (DDPG)

$$\operatorname{argmax} Q^\theta \leftrightarrows L(\theta)$$
$$\max Q^\theta \leftrightarrows \pi$$

Given that we have **Discrete** actions, we can enumerate them.

What if the action is not discrete? [Lillicrap et al., 2015]

Parameterized Policy Function

$$a^t \leftarrow \pi(o^t) = \boxed{\mu^\Theta(o^t)} \approx \underset{a'}{\operatorname{argmax}}\, Q^\theta(o^t, a')$$

Policy Performance

$$J(\Theta) = \frac{1}{|X|} \sum_x^X Q^\theta\left(o_x^t, \mu^\Theta(o_x^t)\right)$$

Reinforcement Learning Formalized IV

Stochastic Policy Gradient I

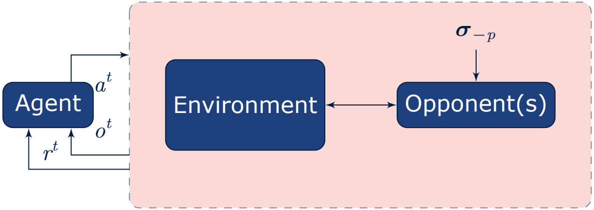
- Actor-Critic Methods
- Stochastic
- On-Policy
- Policy Iteration
- Value Method
- Model-Free

Improvements

SPG algorithm are improved with GAE [Schulman et al., 2016], TRPO [Schulman et al., 2015], and PPO [Schulman et al., 2017].

RL Optimization Problem

$$\text{optimize } \pi(o^t) \mapsto a^t$$
$$\text{s.t. } \max \mathbb{E}\left[\sum_{\tau=0}^{\infty} \gamma^{\tau} \cdot r^{t+\tau} \,\middle|\, \pi, \boldsymbol{\sigma}_{-p}\right]$$



Reinforcement Learning Formalized V

Stochastic Policy Gradient II

The idea of SPG is to update policy to increase the probability of actions with a positive advantage.

Stochastic Actor

$$a^t \sim \pi^{\Theta}(o^t)$$

Value Function

$$V_\pi(o^t) = \mathbb{E}\left[\sum_{\tau=0}^{\infty} \gamma^\tau \cdot r^{t+\tau} \,\middle|\, \pi\right]$$

Advantage

$$A(o^t, a^t) = r^t + \gamma \cdot V_\pi(o^{t+1}) - V(o^t)$$

Policy Performance

$$J(\Theta) = \frac{1}{|X|}\sum_{x}^{X} \boxed{\log \pi^{\Theta}(a_x^t | o_x^t)} \cdot A(o_x^t, a_x^t)$$

Value Function Loss

$$L(\theta) = \frac{1}{|X|}\sum_{x}^{X} \left(r_x^t + \gamma \cdot V_\pi(o_x^{t+1}) - V_\pi(o_x^t)\right)^2$$

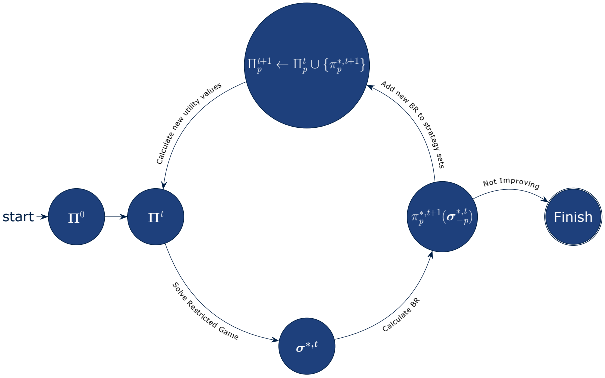
Experience

$$x = \langle o_x^t, a_x^t, o_x^{t+1}, r_x^t \rangle \in X \sim \mathfrak{T}$$

Policy Space Response Oracles

PennState
College of Information
Sciences and Technology

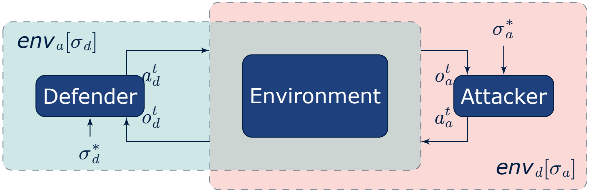
Policy Space Response Oracles



Require: Initial strategy sets $\mathbf{\Pi}$;
Compute *Expected Utilities* $U^{\boldsymbol{\pi}}$ for each strategy profile $\boldsymbol{\pi} \in \mathbf{\Pi}$;
Compute MSNE of $\mathbf{\Pi}$ as $\boldsymbol{\sigma}^*$;
for *many epochs* **do**
  for *each player* p **do**
    for *many episodes* **do**
      Sample $\boldsymbol{\pi}^*_{-p} \sim \boldsymbol{\sigma}^*_{-p}$;
      Train $\pi_p^+(\boldsymbol{\pi}_{-p})$ using InRL;
    end
    $\Pi_p \leftarrow \Pi_p \cup \{\pi_p^+\}$;
  end
  Compute $U^{\boldsymbol{\pi}}$ for new strategies ;
  Compute MSNE of $\mathbf{\Pi}$ as $\boldsymbol{\sigma}^*$;
end
Output current solution $\boldsymbol{\sigma}^*$;

PSRO algorithm [Lanctot et al., 2017] based on Double Oracles [McMahan et al., 2003].

Policy Space Response Oracles



Result: set of pure policies Π^a and Π^d
$\Pi_a \leftarrow$ attacker heuristics;
$\Pi_d \leftarrow$ defener heuristics;
while $U_p(\sigma_p, \sigma_{-p})$ *not converged* **do**
  $\sigma_a, \sigma_d \leftarrow$ `solve_MSNE`(Π_a, Π_d);
  $\theta \leftarrow$ `random`;
  $\pi_a^+ \leftarrow$ `train`$(T \cdot N_e, env_a[\sigma_d], \theta)$;
  $\Pi_a \leftarrow \Pi_a \cup \pi_a^+$;
  assess π_a^+;
  $\sigma_a, \sigma_d \leftarrow$ solve_MSNE(Π_a, Π_d);
  $\theta \leftarrow$ `random`;
  $\pi_d^+ \leftarrow$ `train`$(T \cdot N_e, env_d[\sigma_a], \theta)$;
  $\Pi_d \leftarrow \Pi_d \cup \pi_d^+$;
  assess π_d^+;
end