Resilient Decision Making in Adversarial and Uncertain Environment

By

Yi Li

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

December 14th, 2019

Nashville, Tennessee

Approved:

Yevgeniy Vorobeychik , Ph.D.

Xenofon Koutsoukos, Ph.D.

Gautam Biswas, Ph.D.

Abhishek Dubey, Ph.D.

Zhijun Yin, Ph.D.

ABSTRACT

Benefiting from the recent progress of AI research, computerized supports in making decisions have been deployed to a large variety of computer systems supporting decision making, which received research interests in the area of decision support programs, systems, methods, and techniques. However, recent cases started to reveal the decision support vulnerabilities, where human intervention becomes infeasible to handle problems such as failures or deliberate attacks. On the other hand, such situations are not despairing as tremendous approaches addressing resilience and robustness have been explored by the theoretical side, which leads to massive success in many areas.

This dissertation aims to bridge this gap to improve the resilience of decision support systems in several different cases. We first consider the scenario of decentralized traffic light control problems and develop a cloud computing framework addressing simulation-based optimizations driven by real-time data. We further study the multiple autonomous vehicles path planning with motion uncertainty, model the interactions among strategic agents via a game-theoretical framework, and investigate the gap between centralized and decentralized control scheme. Then, we focus on the areas with machine learning enhanced decision support systems (medical imaging and anomaly detection). The vulnerabilities of such systems are revealed and addressed by our resilient algorithm. Finally, we present a game-theoretical approach addressing the threaten of adversarial examples in machine learning enhanced systems by calculating the optimal randomization scheme over multiple learning models.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Chapter 1

Introduction

1.1 Background

Since the first general theory of decision process was established by Nicolas of Condorcet (1743-1794), a great enlightenment philosopher, as part of his motivation for the French constitution of 1793, decision theory has received undivided attentions due to its abstractness on problem-solving methodology. Modern decision theory, as a truly interdisciplinary subject of its own right, has dramatically developed since the 1950s and been pursued by researchers with different backgrounds, like economists, statisticians, psychologists, social scientists, and philosophers. Benefiting from the recent progress of computer science, computerized supports in making decisions have been deployed to a large variety of computer systems supporting decision making, which leads computer scientists to be a new member of the team and turns the field to be at another turning point of possibilities.

In the early applications of computerized decision supports, the system usually can be considered as a framework integrating a collection of data, models, information processing, and other expertise, which provides a unified way for analysis of problems and evaluation of the optimal (sub-optimal) solution from the set of alternatives. Such a structure provides the capability of combining various computer technologies to solve complex and large scale problems. One typical example is FORPLAN [54], which is a linear programming system used to support national forest land management planning. FORPLAN was designed to have flexibility in importing and exporting data (e.g., supporting different kinds of land organization importing from geographic information systems), optimization or simulation backends, and user presentation modules (e.g., the compatibility with Forest Service accounting systems), leading a great success in nature resource management at a national level.

For solving those problems with well-defined structures, it is not surprising that the computerized decision supports can help due to the reason that the corresponding optimization models and solution methods coming from the outcome of modern decision theory allow the extra computing power to be used in a systematic way. However, such optimization-leaded patterns are unrealistic in many cases where targeted problems are unstructured or unquantifiable. Moreover, the poor portability of models among problems (model formulations are usually dependent on human expertise with domain-specific knowledge) leads the development process to be expensive and difficult to be practiced in an incremental way. Therefore, the idea for seeking a ”general problem solver” emulating human reasoning activities, independent of subject matter, introduces the ”intelligence” to the field of computerized decision supports.

Starting at the 1970s, it became apparent that intelligent decision support systems, based on the AI technologies falling into the category of symbolic reasoning, widely emerged, including Expert Systems, Knowledge-based Systems, Semantic reasoner, Frame-Based Systems and Truth Maintenance System. The idea behind these is to provide an abstract layer between an inference engine and inference rules. While an inference engine is the core component of decision making being able to infer logical consequences from a set of asserted general facts or axioms, inference rules capture knowledge about a specific domain in some structured symbolic form in a particular representation (e.g., Semantic web), created by domain experts. Therefore, such approaches have the capability of representing heuristic knowledge symbolically and manipulate them in an automated way via reasoning.

Another common type of AI methods that of machine learning has been deployed for decision support. In the aspect of decision support system, the strength of machine learning is the capability of ”meta-modeling”, which aims to address unstructured problems, being complementary to reasoning-based approaches assuming a specific type of relationship between inputs and output. In this way, we can create a machine learning enhanced decision-making process exhibiting very complex behavior by learning it from data without being explicitly programming.

This thesis investigates scenarios in which decision-makers can be enhanced and receive the capability to make resilient decisions under uncertain and adversarial environments. In this chapter, we first attempt to demonstrate the motivations for the work discussed in this thesis. Then, we state the thesis questions by illustrating an ideal application scenario. Finally, we present our contributions, followed by a synopsis of the thesis contents chapter-by-chapter.

1.2 Motivation

Computerized support in making decisions has been deployed in various critical cyber-physical infrastructures, such as the national power grid, transportation network, smart cities, are large-scale and intricate systems that illustrate highly dynamic and uncertain operations, significant heterogeneity in the end systems, network protocols and technologies, as well as software systems that support the system operations, which arouses the research interests in the area of decision support programs, systems, methods and techniques.

While most efforts are focusing on fast, precise and effective decision-makers, decision support vulnerabilities are being revealed in many different cases, where human intervention becomes infeasible to handle problems such as failures or deliberate attacks. For example,

- In December 2015, an attack reported on Ukraine power grid lead the disconnections of seven 110kV and three 35kV substations and resulting in a power outage for 80k people for three hours.

- In December 2016, Yahoo announced that 500 million user accounts had been hacked in 2014 and a different attack in 2013 compromised more than 1 billion accounts.

- In May 2017, it was reported that cyber attackers had accessed Target's gateway server and stole data from up to 40 million credit and debit cards of shoppers.

It is foreseeable that the number of threat posed adversaries will continue to increase as more and more subtle vulnerabilities are introduced by innovative technologies.

Malicious attacks are not the only challenges. Due to the lack of perfect sensors and deterministic observations, in the real world, decision-makers usually have to deal uncertain knowledge about the surrounding environment and can be further exploited by misinformation due to attacks. For instance, an autonomous vehicle uses a low-accuracy GPS for localization, and the motion planner not taking that into account may yield a dangerous trajectory that collides with obstacles. Such case can be further complicated, if the autonomous vehicle considers the uncertainty caused by other traffic participants (e.g., a non-autonomous vehicle). As a result, the capability for dealing with uncertainty constraints and malicious attacks is an urgent demand in computerized decision support systems.

1.3 Key Research Challenges

In this thesis, the term ”resilience” describes how systems operate at an acceptable level of normalcy despite disturbances or threats, providing a conceptual idea of what a better decision-support system looks like. Despite retained common traits, resilience takes various domain-specific meanings in different disciplines. In this section, we demonstrate a serial of research challenges which this thesis focuses on, deriving from the domain-specific definitions of resilience specialized in scenarios of various decision support systems.

- In the first challenge, resilience refers to the capability of real-time computing and

1.3.1 Challenge 1: How can real-time decision making be addressed by the simulation-based optimizations driven by real-time data?

To provide high-quality decision support, one can use simulations in an optimization loop to derive the best values of system parameters for a given system state, particularly when the system has too many parameters and traditional means to optimize the outcomes are intractable. To that end, simulation-based optimization methods have emerged to enable optimization in the context of complex, black-box simulations, thereby obviating the need for specific and accurate model information, such as gradient computation. A significant challenge in using simulation-based optimization is optimizing the decision parameters for real-time systems.

Concretely, suppose a decentralized feedback traffic light control scenario that aims to improve the overall traffic performance (minimize congestion) of a certain area by manipulating the behaviors of traffic lights. In this scenario, to yield the control parameters with high quality via simulation-based optimization, there are two key challenges. The first is the close combination with data-acquisition technology providing the real-time road information (e.g., vehicle flows) such that the simulation results have the capabilities to make fast responses to the emergent events (e.g., road accident). Moreover, to ensure scalability and real-time decision support, one must be able to rapidly deploy simulation-based optimization in a way that makes the best use of available computing resources given the time and budget constraints.

1.3.2 Challenge 2: In multi-agent path planning, how does the interaction among agents with motion uncertainty affect the outcomes?

Path planning is a fundamental decision-making problem in autonomous robotic control. In much of the research on path planning, including mobile robot navigation, a fundamental task is to find a resilient control sequence which yields a collision-free motion from a starting position to the goal position given a collection of known obstacles. As interactions among autonomous vehicles, be it on our roads or in the skies, becomes more routine, we can expect a certain amount of conflict to emerge, as the autonomous agents, designed in service of their individual goals, must occasionally find these goals dependent on other autonomous agents nearby. However, it is natural to arise the question of what autonomous vehicle ecosystem would thereby emerge, when many autonomous agents attempt to achieve their individual goals, but must necessarily interact with one another in doing so.

1.3.3 Challenge 3: How to improve the robustness of machine learning enhanced decision support system?

Machine learning is increasingly used to enhance decision support by automating the processing of large datasets. In particular, the machine learning based imaging recognition algorithms are extensively used to find the solutions to various challenges arising in the many fields (e.g., medical imaging [34, 43, 9, 30] and cyber-physical systems [104, 67, 74]). In these cases, the accuracy and reliability of the prediction model are the vital properties, since any mistake make can be disastrous and cost human lives.

Despite advancements in machine learning algorithms, it has been shown that machine systems are inherently vulnerable to carefully-crafted attacks. In such cases, attackers can exploit the model's weaknesses at the testing phase by crafting malicious adversarial examples producing intentional errors, and evade detections [12, 22, 45]. The intuition behind evasion attacks is adding a tiny amount of well-tuned additive perturbation to the original data input and significantly changing the prediction in the testing time(e.g., leading a classifier to label the modified image as a completely different class). Such an issue becomes one of the key challenges of this work.

1.3.4 Challenge 4: How to forge a resilient anomaly detector in sensor networks?

Cyber-physical systems feature a control loop that maps sensor measurements to control decisions that involve maintaining system state features, such as flow speed, temperature, and pressure, in a safe range. In these cases, perturbations of normal behavior indicate a presence of intended or unintended included attacks, defects, faults, and so on. Thus, anomaly detection can be a key for solving potential intrusions being those activities that can change the system behaviors, with anomaly detection employed to ensure that anomalous or malicious sensor measurements do not subvert system operation. A recent tendency to use machine learning for anomaly detection significantly helps in enhancing the performance of detection (in both speed and quality).

Nevertheless, even armed with anomaly detectors, systems can be vulnerable to stealthy attacks [37], whereby an attacker submits measurements of compromised sensors in a way to ensure that they appear normal while influencing the behavior of the control loop. To further investigate and quantify such vulnerability becomes another challenge.

1.4 Contributions of the Thesis

This section highlights our research contribution, including framework design and theoretical models design. All of these will be discussed in more detail in the Conclusions chapter.

1.4.1 Addressing Challenge 1: Simulation-based optimization as a service

We have developed a cloud-based framework that provides a *simulation-based optimization as a service* (SBOaaS), in which real-time considerations are explicitly accounted for making optimal use of limited but parallel computational resources in order to obtain the best answer within the given time constraints. This part of the work focuses on presenting a generic optimization process for deploying simulation-based optimization on a cloud architecture. Our framework consists of

- The implementation of SBOaaS, which for a given optimization problem, describes how to decompose the input problem into a group of parallel simulations and efficiently use the existing computing power.

- An anytime parallel simulation-based optimization approach, which admits significant flexibility in both time and computational resource constraints to obtain the best (but possibly sub-optimal) solutions given the available resources and time constraints on decisions.

This work has resulted in the following publication.

Li, Y., Shekhar, S., Vorobeychik, Y., Koutsoukos, X., Gokhale, A. (2018). Simulation-Based Optimization as a Service for Dynamic Data-Driven Applications Systems. *In Handbook of Dynamic Data Driven Applications Systems* (pp. 589-614). Springer, Cham.

1.4.2 Addressing Challenge 2: Path planning games

To investigate the consequences of such strategic interactions among multiple path planners and how these planners make resilient decisions, we propose a study of path planning games. An important feature of such games is that a collection of self-interested path planners each trade-off two objectives: efficiency, or speed with which their goals are achieved, and safety, or probability that they crash before reaching their goals. The highlight of the work includes

- Modeling agent's motion dynamic with motion uncertainty.
- Both decentralized and centralized multi-agent path planer via conjunctive optimizations.
- Empirical work on investigating the gap between equilibrium and global optimum.

This work has resulted in the following publication.

Li, Y., Vorobeychik, Y. (2018). Path Planning Games. *International Workshop on Optimisation in Multi-Agent Systems* (OPTMAS) 2018.

1.4.3 Addressing Challenge 3: Robust medical imaging and feature selection games

We first consider a robust imaging recognition problem in the field of medical imaging. We show that medical imaging applications are susceptible to state-of-the-art adversarial attacks, and such vulnerability can be addressed via integrated deep learning models. Specifically, we use brain age prediction as an application to show the following

- A Single visibly imperceptible noise field can dramatically reduce task accuracy.
- A general visibly imperceptible noise field can be generated which will reduce task accuracy for a large batch of subjects, and
- Such adversarial perturbations have significantly less impact on deep learning models that use anatomical context.

Then, we further study the approach addressing the vulnerability of machine learning enhanced decision supports systems by modeling the interaction between the attacker and the learner via *feature selection games*, which includes

- A game theoretical approach finding the optimal feature subset maximizing the robust accuracy in the cases where the attacker can change a limited number of features.
- The comparison to state of the art defenses approaches.

This work has resulted in the following publication.

Li, Y., Zhang, H., Bermudez, C., Chen, Y., Landman, B. A., Vorobeychik, Y. (2019). Anatomical Context Protects Deep Learning from Adversarial Perturbations in Medical Imaging. *Neurocomputing*.

1.4.4 Addressing Challenge 4: Adversarial Gaussian process regression in sensor networks

We consider the problem of vulnerability of CPS with GPR-based anomaly detection to stealthy attacks, as well as the corresponding problem of making such systems robust, which includes the following contributions

- A model of Gaussian process regression based anomaly detection for cyber psychical systems.
- A novel approach to find the optimal stealthy attack penetrating detections.
- A game theoretical framework, in which the defender considers sensor selection, in addition to the choice of detection thresholds, as a lever for making anomaly detection more robust to attacks.

1.4.5 Dissertation Outline

The remainder of the dissertation is organized as follows: Chapter 2 presents prior work that relates to our research; Chapter 3 describes our approach to integrate simulation-based optimization as a service to traffic light control problem; Chapter 4 describes the path planning game; Chapter 5 presents the formulation of resilient medical imaging recognition; Chapter 6 explores the idea of feature selection games; Chapter 7 discusses adversarial Gaussian process regression; and Chapter 8 is the conclusion.

Chapter 2

Related Works

2.1 Cloud-based Services for Simulations and DDDAS Applications

mJADES [91] is a Java-based simulation engine that can automatically acquire resources from various cloud providers and perform simulations on virtual machines. This approach is similar to ours in spawning simulations, however, the objective is different and it does not provide aggregation-based optimization logic. DEXSim [28] is another simulation framework based on distributed systems principles that can provide two-level parallelism by accounting for CPU threads and availability of multiple systems. On the other hand, SBOaaS relies on the Linux kernel for scheduling of container processes to avail multiple CPU cores on the physical server. Another cloud middleware is the RESTful interoperability simulation environment (RISE) [1] which applies RESTful services for remote management of simulation server using Android-based hand held devices.

Resilient DDDAS-As-A Service (rDaaS) [5] is a cloud-based trustworthy and resilient infrastructure for developing secure crisis management systems using DDDAS principles of instrumentation, continuous monitoring and adaptation. The rDaaS architecture's goal is to align the cloud technology required for providing crisis management system in accordance with DDDAS paradigm by combining the design and runtime stages. Similar to rDaaS, SBOaaS (in Chapter 3) leverages the DDDAS paradigm to provide cloud services, however, the objective of SBOaaS is to solve optimization problems using cloud based simulations and the methodology requires managing cloud resources at scale as the number of application instances required by SBOaaS is much larger compared to rDaaS. Nguyen and Khan [80] describe a framework for supporting DDDAS applications in cloud that proactively performs resource optimization and allocates resources when the sampling rate of the DDDAS application changes. This work does not consider the virtualization layer present on the cloud servers and the effects of co-location of multiple different jobs, in contrast, SBOaaS considers the virtualizaton overhead imposed by Docker containers and optimizes the resources for all the scheduled jobs.

2.2 Traffic Light Optimal Control Problem

Fundamentally, a traffic light control problem is a form of scheduling problems for switching control actions on stochastic hybrid systems. Various models have been well-studied. A decision tree model with Rolling Horizon Dynamic Programming was presented by Porche [90]. The approach based on multi-objective Maxed Integer Linear Programming formulation was proposed by Dujardin [33]. A Markov Decision Process approach was proposed in Yu and Recker [120] and Reinforcement Learning was used in Thorpe [110]. Choi [29] implemented a first-order Sugeno fuzzy model and integrated it into a fuzzy logic controller, while an Infinitesimal Perturbation Analysis approach, using a Stochastic Flow Model to represent the queue content dynamic of road at an intersection was presented by Panayiotou [84].

However, to find the optimal control parameters for a traffic light system via close loop simulations is still a big challenge due to its high computational complexity and the requirement on real time reactions. In this work, we illustrate that SBOaaS is a suitable framework to address such issues.

2.3 Multi-Agent Path Planning

One common paradigm for studying multi-agent path planning problems is by considering cooperative path planning involving multiple agents. For example, Shen et al. [103] studied cooperative path planning in UAV control system, while LaValle [65] presented an algorithm for applying path planning with stochastic optimal control.

Game theoretic problems related to path planning have been considered from several perspectives. Closest to traditional path planning are zero-sum models of games against nature in which agents are designed to be robust against adversarial uncertainty in the environment [25, 24]. Classic approaches consider rules of interaction and negotiation among self-interested agents, including planning agents [93, 50, 51]. Loosely related also is the extensive literature on multi-agent learning, in which multiple agents repeatedly interact in strategic scenarios in which rewards and dynamics depend on all agents (often modeled as stochastic games) [105].

The game theoretical framework, as a powerful tool for solving and analyzing the path problems in multi-agents systems under both reciprocal and adversarial environments, becomes more and more popular in recent years. Many non-cooperative multi-agent problem are formulated as the framework, where a set of heterogeneous agents share a common limited resource, aim to achieve their tasks, and avoid to conflicts by negotiating with each other. Here the decisions making for each agent depends on not only its own actions but also on the actions of the others. [93] discuss the different patterns of rules of interaction and negotiation among agents and how these designs of rules can achieve some desirable properties for the whole group of agents, such as stability and efficiency. [105] gives a survey about the process that self-interested agents can adapt behavior to changing circumstances via learning technique.

On the other hand, the multi-agents path planning is a typical multi agent planning problem, where agents aim to find the path to reach their goal positions and avoid collisions under limited spatial resource. Many recent researches focus on the cases with cooperation. [103] studied the cooperative and intelligent path planning in UAV control system, while [65] presented an algorithm for applying path planning with stochastic optimal control and multiplayer games. Another popular topic is various imaginary path planning problem in games on a plane, like reach avoid game [25], [24]. However, as far as we know, the path planning problem in non-zero-sum non-cooperative cases have not been deeply considered by any existing work.

2.4 Routing Games

Another important class of game theoretic models related to path planning are *routing games*. The routing games, as a framework for modeling routing traffic in a large communication network, were first informally discussed by Pigou [88]. This model was first formally defined by Wardrop [116] based on a flow network under the non-atomicity assumption. Therefore, equilibrium flows in non-atomic selfish routing games are often called *Wardrop equilibria*. Since then, a number of fundamental results for the non-atomic routing games have been proved by various researchers, such as the existence and uniqueness of equilibrium flows [7], first-order conditions for convex programming problem [11], and the theory of general non-cooperative non-atomic games [99]. The seminal work by Roughgarden and Tardos [95] first characterized the gap between centralized and decentralized control in multi-agent routing problems, formalized as the price of anarchy, or ratio of socially optimal to worst-case equilibrium outcomes. Their work explained the principles behind a broad class of counter-intuitive phenomena, such as Braess's Paradox [14].

Both routing games and path planning games (in Chapter 4) investigate the competition among agents during their navigation tasks (e.g. passing through bottlenecks). However, in routing games, the state space is a graph-based structure, and the cost of competition is modeled by a set of latency functions without considering the agents' dynamics, while path planning games consider the problem at higher fidelity, with a continuous state space where the latency is caused by the interaction among agents. Moreover, our model of path planning games allows us to explicitly study the tradeoff agents make between performance and safety, an issue not considered in routing games.

2.5 MDP approaches

Due to the lack of perfect sensors and deterministic actions, in real world, stochastic systems are faced with various kinds of uncertainty, which typically have a chance of failure caused by unexpected events which affect agents' motion. In this case, planning under motion uncertainty is a particularly challenging problem. To reduce the chance of collision, an agent will try to avoid those unsafe states, such as staying at the initial placement, which usually reduces mission performance since it diatribes the robot to reach to its goal position. One popular approach aiming to address the gap between risk and performance is to assign a positive reward for reaching the goal and a negative reward for collisions so that the path planning problem can be formulated by a Markov Decision Process encoding. This approach is usually implemented under some (discretized) map space with finite states. To find the optimal path, dynamic programming derived from Bellman equation is usually considered, due to its guaranteed soundness and fast rate of convergence. The surveys of the MDP method for path planning can be found in [46] and [64]. However, due to exponential growing cost for discretizing in high dimensional space, such approach is usually in the cases with multi-agent and complicate action space.

Compared to MDP approaches, path planning games are modeled under the continuous state space without discretization, which makes path planning games have the capability to handle multi-agent path planning problem in high-dimension space (e.g. configuration space path planning). Moreover, in many practical cases, it is difficult to define the value of mission and failure reward for MDP approach due to the lack of realistic meaning, while path planning games take into account those real world positive and negative factors (time step consumed by agents to reach its target, safety margin)

2.6 Gaussian Process Regression

Gaussian Process regression (GPR) [117] is a popular choice for anomaly detection systems due to its power and flexibility [27, 69, 36, 2].

Neural Networks (NN) have been used in many control systems due to the avoidance of the limitations of approximated models [79]. Recently, Gaussian Process methods raised [117] in this field, as both GPs and NNs have the universal approximation capabilities and

GPs outperform NNs under certain conditions [40]. The community of anomaly detection shows the particular interest on GPs, due to its nonlinear, nonparametric preperties [27, 69, 2]. GPs also provide uncertainty estimates for their prediction, which is important for construct the bias-free anomaly identification methods.

2.7 Stackelberg Security Game

Stackelberg Security Game (SSG) was first introduced by Kiekintveld et al., representing specializations of a particular type of Stackelberg game [113], where a defender (the leader) defends a set of targets using a limited number of resources and an attacker (the follower) attack with the observation of the defender's strategy. The solution to a SSG is a mixed strategy for the defender maximizing the expected utility, is known as a Stackelberg equilibrium [68], where no player has the incentive to deviate. The strong Stackelberg equilibrium (SSE) is the most commonly adopted version of Stackelberg equilibrium, assuming that the attacker always breaks ties by choosing the best action for the defender [15, 31, 81, 114]. As an SSE exists in all Stackelberg, it is an attractive solution concept compared to Stackelberg equilibrium with other tie-breaking rules. However, there are some newly proposed solution concepts that are more robust against various uncertainties and have been used in later applications [119, 3, 89]. As Conitzer and Sandholm first presented the algorithms for computing optimal commitment strategies in Bayesian Stackelberg games [31], an improved algorithm called DOBSS [81] is deployed to ARMOR, a fielded application in use at the Los Angeles International Airport [48]. Another algorithm called ASPEN[47] that is designed to compute SSE in domains with a very large number of pure strategies for the defender have been deployed in IRIS system (Intelligent Randomization In Scheduling) by FAMS (The US Federal Air Marshals Service) since 2009 to randomize schedules of air marshals on international flights.

Our work in Chapter 7 looked at in cyber-security using the SSG framework to design robust anomaly detection systems. Other recent cyber-security applications based on the

SSG framework include inspecting a large number of alerts from any intrusion prevention system [98] and fighting against spear phishing [122].

2.8 Adversarial Machine Learning

Since Szegedy et al. first concerned with the stability of neural networks with respect to small perturbations to their input and discovered the existence of adversarial examples which are crafted by applying undetectable perturbations to the original input and can force a well-performing deep neural networks to produce incorrect outputs, a lot of attention has been paid to the context of adversarial learning and the security of deep neural networks [23, 121, 60, 38, 85]. On the other hand, a number of recent works have been proposed to mitigate the effects of adversarial attacks, including adversarial training [111], distillation [87] and approaches based on Generative Adversarial Network [97].

Our work in Chapter 5 considered defense for a robust imaging recognition problem in the field of medical imaging via domain-specific context information. The work in Chapter 6 presented a game-theoretical approach to calculate the learner's optimal strategy (other game-theoretical approaches for addressing adversarial examples are detailed in [123]). In Chapter 7, we considered an adversarial setting on machine learning enhanced anomaly detector with a specific structure that requires a novel solution approach to address.

2.9 Anomaly Detection

The existing works on anomaly detection in CPS include several different models, with Gaussian Process regression (GPR) [117] a popular choice due to its power and flexibility [27, 69, 36, 2]. However, other approaches based on machine learning have also been explored [78, 53]. An important strand in this literature also involves leveraging physical models in anomaly detection [19, 112]. Cárdenas et al. [19] [19] studied the use of physical models for anomaly detection. However, most of this work does not consider attacks vulnerability of anomaly detection to stealthy attacks. Similarly, there is an extensive literature on the problem of sensor selection in non-adversarial settings [52, 59, 101]. The extension to consider adversarial sensor selection has received some attention [58, 63], but focuses largely on robustness to denial-of-service attacks on sensors, rather than the integrity attacks that we consider. An important precursor to our work is Ghafouri et al. [37] [37], who consider robust anomaly detection in the context of stealthy integrity attacks. However, this work considers conventional regression which yields deterministic predictions of sensor values, in contrast to GPR, where prediction is a random variable, adding a non-trivial technical challenge to the problem of stealthy attacks. Moreover, Ghafouri et al. do not consider the problem of sensor selection, focusing solely on tuning anomaly detection thresholds. As our experiments demonstrate (in Chapter 7), the ability to select sensors accounts for *most* of the robustness in our setting.

Chapter 3

Simulation-based Optimization as a Service

3.1 Problem Overview

Dynamic data-driven applications systems (DDDAS) must be adaptive in the face of highly fluctuating and uncertain environments. An important means to such adaptability is through the use of simulation models which can be leveraged for dynamic decision support. To provide high quality decision support, one can use simulations in an optimization loop to derive the best values of system parameters for a given system state particularly when the system has too many parameters and traditional means to optimize the outcomes are intractable. To that end, simulation-based optimization methods have emerged to enable optimization in the context of complex, black-box simulations thereby obviating the need for specific and accurate model information, such as gradient computation. An important challenge in using simulation-based optimization is optimizing the decision parameters. However, to ensure scalability and real-time decision support, one must be able to rapidly deploy simulation-based optimization in a way that makes the best use of available computing resources given the time and budget constraints. To address these needs, we propose a cloud-based framework for simulation-based optimization as a service (SBOaaS) to enable a flexible and highly parallelizable dynamic decision support for such environments. We illustrate the framework by using it to design a dynamic traffic light control system through simulation-based optimizations using the Simulation of Urban Mobility (SUMO) traffic simulation model that adjusts to the observed vehicle flow.

The rest of the chapter is organized as follows: Section 3.2 provides an overview of our simulation-based optimization as a service concept; Section 3.3 describes the algorithms behind realizing SBOaaS particularly in the context of Anytime computations; Section 3.4 describes the system architecture we have developed to deploy SBOaaS; Section 3.5 validates our claims; and finally Section 3.6 describes concluding remarks alluding to future challenges.

3.2 Overview of SBOaaS

In this sectionwe use a motivational case study to develop the problem statement we have formulated and solved in this paper. To that end we first present a traffic light control system as an example of a real-world system where high-quality configuration of the traffic light controller requires an iterative black-box optimization process based on data-driven model simulations. Owing to the high demand for resources and real time performance constraints, such a capability requires cloud computing resources. We designed and implemented SBOaaS, a framework for simulation-based optimization as a service. This section presents key features and a case study illustrating those challenges that SBOaaS should address.

3.2.1 Motivating Case Study: Dynamic Traffic Light Control System

To formulate the problem statement, we use a dynamic traffic light control scenario as our motivating example. In this scenario, each intersection traffic light controller switches its traffic light phases according to the observed vehicle flow. In general, a traffic light phase is related to a collection of lanes dominated by such a phase; if the number of waiting vehicles in the lanes related to the current phase is small and the number of waiting vehicles in the lanes related to the next phase is large, the controller will switch the traffic light phase. Figure 3.1 provides a visual demonstration of the controller logic.

Formally, a feedback controller has a predefined phase sequence $(p_0, ..., p_n)$. For each phase p_i, m_i is the minimum interval, M_i is the maximal interval, q_i is the average queue length of the lanes related to the i^{th} phase, and θ_i is the threshold on the queue length of lanes blocked in the i^{th} phase. If t is the current time point, the control logic is as depicted in Algorithm 1.

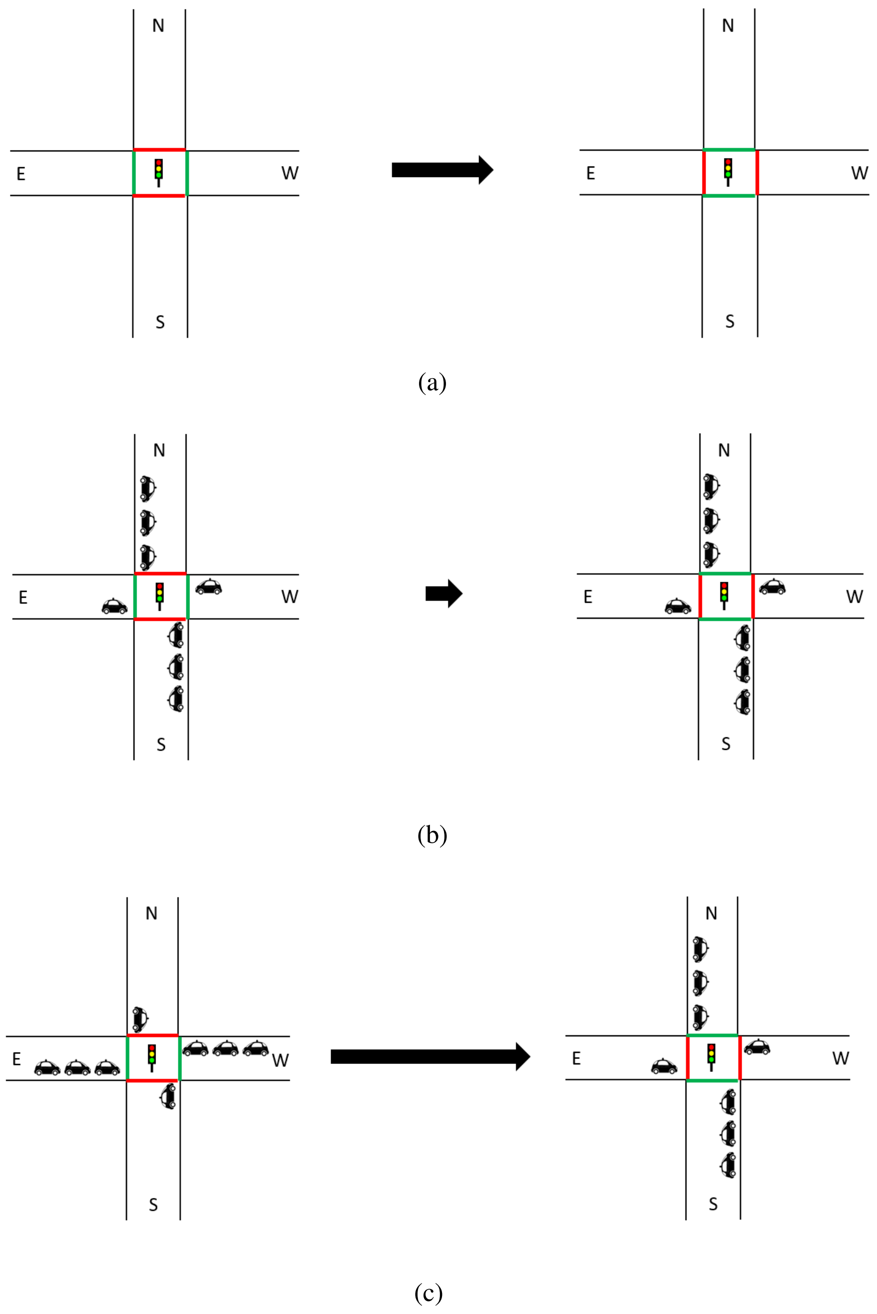


Figure 3.1: The control logic for feedback controllers. (a) Non-feedback controllers have a fixed interval between two phases. (b,c) Feedback controllers dynamically change the interval according to the length of their vehicle queues.

The controller must solve an optimization problem as follows: for a given vehicle flow of an area in a certain time period and a set of controlled intersections $I\{I_0,...,I_m\}$, find the optimal thresholds $(\Theta_0,...,\Theta_m)$, where $\Theta_i = (\theta_0,...\theta_{n_i})$ are the thresholds of the ith

Algorithm 1 Feedback Controller

```
1: Current Phase $P := p_0, t' := t, i := 0$.
2: loop
3:     $i_{next} := (i+1) \mod n$
4:     if $t - t' > m_i$ then
5:         if Reach to the maximum interval, $t - t' = M_i$ then
6:             Switch phase, $P = p_{i_{next}}, i = i_{next}$
7:         else if Find the congestion, $q_i < \theta_i, q_{i_{next}} \geq \theta_{i_{next}}$ then
8:             Switch phase, $P = p_{i_{next}}, i = i_{next}$
9:         end if
10:     end if
11: end loop
```

intersection.

The scenario with a single intersection with similar control logic has been discussed in many prior research efforts, e.g., [71]. However, the situation becomes much more complicated when generalizing the controller model to cases with multiple intersections and correspondingly multiple traffic lights. Many factors, such as densities of vehicle flows and topological structures of road networks, may affect the outcomes of such road systems, which leads to the issue of defining the model describing the interactions among the intersections.

3.2.2 DDDAS-specific Problem Statement and the SBOaaS Approach

Examples, such as the traffic light for multiple intersections, say, in a city downtown, pose significant challenges due to the compute-intensive nature of the solution approach. Moreover, the dynamic nature of traffic patterns (e.g., morning and evening rush hour versus afternoon and night hours) will require periodically recomputing the optimal parameters, which further complicates the problem and its demands on resources.

Two fundamental problems exist in this realm. First, it is likely that the DDDAS feedback loop may have access to only black box models of the dynamic systems, yet will require that the DDDAS infrastructure obtain optimal parameters to be used in the

DDDAS feedback loop. Second, the significantly compute intensive nature of the solution approaches makes it infeasible to deploy such model simulations in-house. Rather, there is a need for elastic computing capabilities. Thus, the DDDAS problem we solve in this paper can be posed as: (a) How to obtain the optimal parameters, and (b) How to elastically scale the compute resources as the computational needs of the solution approach dynamically changes?

This paper solves this fundamental problem using the following duo of synergistic approaches: First, we use simulations in an optimization loop to derive the best values of system parameters for a given system state particularly when the system has too many parameters and traditional means to optimize the outcomes are intractable. The approach is called *simulation-based optimization.* To address the need for elastic resources, we exploit Cloud computing as the means to address these needs and provide a framework to realize what we call *Simulation-based Optimization-as-a-Service (SBOaaS).*

Figure 3.2 visually represents how SBOaaS can be used to deploy the dynamic traffic light control system with online simulation-based optimization. The control system is a closed loop, periodically receiving the real time distribution of vehicle flows – which represents the dynamic and data-driven traits of DDDAS – running multiple simulations in parallel to find the optimal thresholds, and sending the feedback to the traffic light controllers – which represents the closing of the loop in DDDAS.

3.2.3 Key Features of SBOaaS

The following represent the key features of SBOaaS.

- **A cloud based solution for parallel execution of multiple simulations.** Applying computationally expensive online simulation-based optimization is usually time consuming and often fails to address the real-time constraints of applications. Moreover, for stochastic simulation models, every simulation process can vary and yield different results. To analyze the temporal properties of a stochastic system, a large

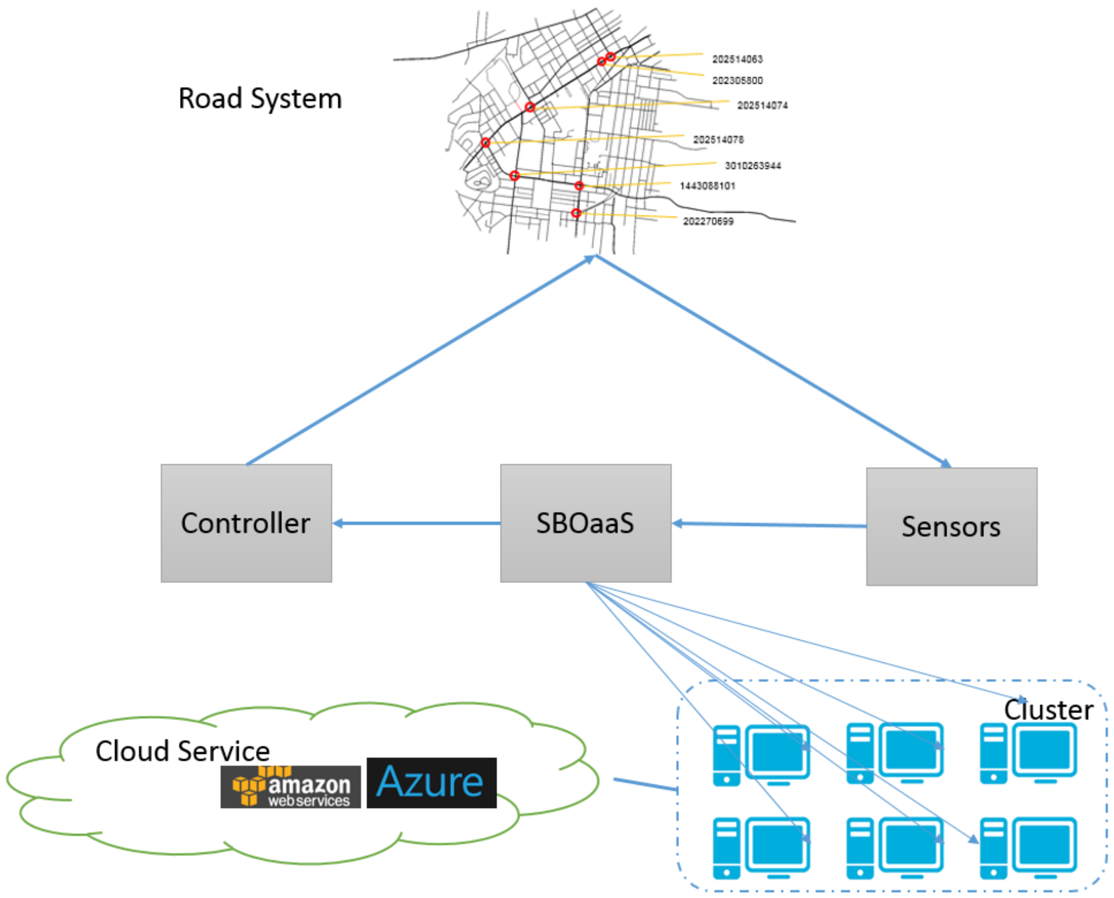


Figure 3.2: SBOaaS for dynamic traffic light control system

number of simulation tasks needs be executed to obtain the probability distribution of simulation results. Thus, the simulation service needs to have the ability to execute multiple simulations in parallel. In our solution, to overcome this problem, we present a cloud-based approach, which is an orchestration middleware helping people to deploy DDDAS applications to the platforms of various cloud service providers without considering platform differences. It integrates the simulation manager having the capability to spawn and execute simulations in parallel and the result aggregation component using several aggregation strategies to recycle the results from the terminated simulations. A web-based interface is also implemented, which allows a user to customize both the simulation model and the input parameters, as well as to monitor the optimization process. Section 3.4 delves into the details of our system architecture.

- **Generic problem decomposition schemes for large scale discrete variable decision problems.** In simulation-based optimization, the results of simulations are often quite different depending on the input parameters supplied to the model. To find the optimal solution, the search space sometimes can be extremely large so that such large-scale problems are intractable to naïve brute force search. In this situation, even parallel computations do not help. In our framework, a collection of generic problem decomposition schemes based on coordinate decent methods is demonstrated, which not only provides an efficient way to parallelize the optimal decision problems with discrete variable domains, but also has the ability to execute anytime optimizations providing a flexible balance between fast response and solution quality.

- **The ability to decouple simulation based problem designs from the problem decomposition schemes.** For traditional model-based online learning and simulation approaches in DDDAS, developers usually need to face and maintain several parts of the system at different levels simultaneously. For example, there is domain-specific knowledge to setup and deploy the simulation environments, different parallelism approaches for various optimization tasks, and system management for regular maintainance. Such a method is not a good practice for a developer team that expects rapid deployment on available resources. SBOaaS leverages Linux container-based infrastructure which aims to create an abstraction layer that helps decouple simulation-based problem designs from the problem decomposition schemes. This approach allows domain experts to encapsulate the simulation environment in a container, while developers design the parallelism process according to the pre-defined interface and system administrators can simply combine both parts to run an optimization without knowing the implementation detail. Moreover, such an approach provides low run-time overhead, negligible setup and tear down costs when deploying the simulations on computing nodes, and fast data exchange among cluster hosts with incremental updates.

3.3 Anytime Optimization Using Parallel Greedy Algorithm

We now describe the approach. SBOaaS estimates the value of an objective, measured using simulation runs, for a given setting of input variables. In order to use this in optimization, one must run this process for many inputs, aiming to choose the best input vector in terms of the objective value. Since such optimization routines can be extremely time consuming in general, they may be of limited utility in dynamic control environments in which real-time decision constraints impose severe limits on the time alloted for simulation-based optimization.

We present several anytime simulation-based optimization algorithms used in our framework which ensure that the optimization process returns the best solution found thus far to the controller even if it is interrupted before it converges. The key feature of these algorithms is that they are directly parallelizable, thereby allowing us to implement them using a cloud-based platform we developed, described below.

Consider a single target optimization problem,

$$\min_{\vec{x}} f(\vec{x}),$$

where $\vec{x} = (x_1, x_2, ..., x_n)$ is a vector of decision variables. In our setting, $f(\vec{x})$ is not known directly, but can be evaluated by simulations for a given $\vec{x}$. If f is stochastic and we wish to minimize the expectation, we can estimate the expectation by running multiple simulations for a given $\vec{x}$ and taking the sample average. Since such a generalization is direct, we assume henceforth that simulations produce a deterministic evaluation of $f(\vec{x})$. We further assume that the domain of variables x_i is discrete. This too is a mild assumption since a continuous, bounded domain can be discretized arbitrarily finely. We developed a framework for anytime simulation-based optimization as a service by making use of a *coordinate greedy* algorithm.

3.3.1 Coordinate Greedy

Coordinate Greedy is a heuristic optimization method which minimizes the function value one variable at a time. The *Sequential coordinate greedy* framework is shown in Algorithm 2.

Algorithm 2 Sequential Coordinate Greedy(f, $\vec{x}^{(0)}$)

```
1: input problem $f$, initial state $\vec{x}^{(0)} = (x_0^{(0)}, ..., x_n^{(0)}) \in \mathbb{R}^n$
2: output $\vec{x}^{(*)} = \arg\min_{\vec{x}} f(\vec{x})$
3: Set $p \leftarrow 0$
4: repeat
5:     for $i \leftarrow 1, ..., n$ do
6:         $x_i^{(p+1)} \leftarrow \arg\min_{x_i} f(x_1^{(p+1)}, ..., x_{i-1}^{(p+1)}, x_i, x_{i+1}^{(p)}, ..., x_n^{(p)})$
7:     end for
8: until termination test satisfied
```

In each iteration, it updates one input variable of f by solving the sub-problem:

$$f_i^{(p)} = \min_x f(x_1^{(p+1)}, ..., x_{i-1}^{(p+1)}, x, x_{i+1}^{(p)}, ..., x_n^{(p)})$$

For the discrete variable domain problem, it converges to the local optimum f^* when there is no further improvement found in one iteration ($\exists P \forall i, f^{(P)}(x_0) = f^{(P)}(x_i)$). Similarly, *stochastic coordinate greedy* selects one variable uniformly at random instead of following the vector order in each iteration. Shalev-Shwartz and Tewari [100] provide the best known convergence bounds for stochastic coordinate greedy.

To parallelize this method, in each step consisting of evaluation of a single component x_i, the framework tries to activate multiple simulations for all possible values in the variable domain. This process keeps running until it is suspended by users or reaches a local optimum, in either case returning the best solution found.

3.3.2 *K*-Coordinate Greedy

With increasing problem dimensionality even the fast coordinate greedy approach becomes expensive. However, limited by the degree of parallelism of basic coordinate greedy, for larger scale problems coordinate greedy does not fully use the provided computing power. ***K*-Coordinate Greedy** is an algorithm that adds another parallelization level to coordinate greedy to accelerate the rate of convergence of the optimization process, as described in Algorithm 3. It initially chooses K, the number of variables to update, according

Algorithm 3 K Coordinate Greedy(f, $\vec{x}^{(0)}$, k)

```
1: input problem $f$, initial state $\vec{x}^{(0)} = (x_0^{(0)}, ..., x_n^{(0)}) \in \mathbb{R}^n$, parallelism degree $k$
2: output $\vec{x}^{(*)} = \arg\min_{\vec{x}} f(\vec{x})$
3: Set $p \leftarrow 0$
4: repeat
5:     Choose index set, $I^{(p)} = \{i_0^{(p)}, i_1^{(p)}, ..., i_k^{(p)}\}$
6:     In parallel on $k$
7:         $x_i^{(p+1)} \leftarrow \arg\min_{x_i} f(x_1^{(p)}, ..., x_i, ..., x_n^{(p)}), i \in I^{(p)}$
8: until termination test satisfied
```

to the available computing resources. In each iteration, it chooses a subset of K variables and optimizes these in parallel using the same update as the coordinate greedy algorithm.

Different parallelism modes do affect the performance and behaviors of K-coordinate greedy. *Synchronous* K-coordinate greedy synchronizes frequently across all K partitions at certain points in time, which ensures that all updates are shared across all processors before further computation occurs, while *asynchronous* one assumes the variable vector x can be accessible to each processor, and available for reading and updating at anytime. Because of eliminating the requirement of consistent information across computing nodes, asynchronous algorithms are supposed to have better performance in practice, while the behaviors of synchronous algorithms are more predictable and easier to analyze. Both synchronous and asyncronous K-coordinate greedy are included and evaluated in our framework.

3.3.3 Adaptive K-Coordinate Greedy

Unlike coordinate greedy, K-coordinate greedy cannot guarantee convergence to the local optimum. The risk of divergence of the algorithm might be increased when there are too many correlated features in the variable vector, which also makes it difficult to define the termination test. In this section, we present adaptive K-coordinate greedy that tries to address these gaps.

Adaptive K-coordinate greedy is a hybrid approach of combining coordinate greedy and K-coordinate greedy, as illustrated by Algorithm 4, which is supposed to speed up the rate of convergence in early stages, and avoid the correlation problems when it gets close to the local optimum. We improve the greedy process by continuously reducing K as the time taken by the algorithm to find the next sub-optimal solution. With decreasing the value of K, the optimization process will be less likely to select correlated features and avoid divergence. When K equals one, the algorithm is exactly the stochastic coordinate greedy algorithm, which has a well-defined termination condition and convergence guarantees.

Algorithm 4 Adaptive K Coordinate Greedy(f, $\vec{x}^{(0)}, k_0$)

```
1:  **input** problem $f$, initial state $\vec{x}^{(0)} = (x_0^{(0)}, ..., x_n^{(0)}) \in \mathbb{R}^n$, initial parallelism degree $k_0$
2:  **output** $\vec{x}^{(*)} = \arg\min_{\vec{x}} f(\vec{x})$
3:  Set $p \leftarrow 0, \triangle t \leftarrow 0$
4:  **repeat**
5:      $k \leftarrow k_0 * exp(-\triangle t / T)$
6:      Choose index set, $I^{(p)} = \{i_0^{(p)}, i_1^{(p)}, ..., i_k^{(p)}\}$
7:      **In parallel** on $k$ processors
8:          $x_i^{(p+1)} \leftarrow \arg\min_{x_i} f(x_1^{(p)}, ..., x_i, ..., x_n^{(p)}), i \in I^{(p)}$
9:      **if** find a better solution **then**
10:         $\triangle t \leftarrow 0$
11:     **else**
12:         Increase $\triangle t$
13:     **end if**
14: **until** termination test satisfied
```

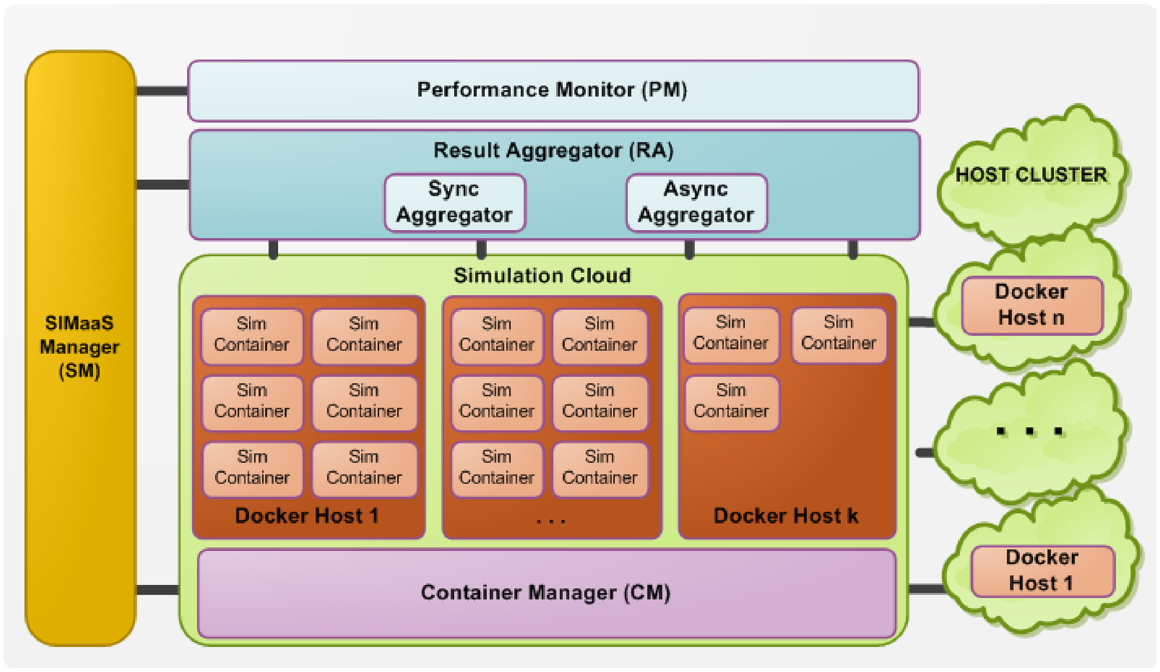


Figure 3.3: System Architecture

3.4 System Architecture

The cloud based SBOaaS architecture is based on our existing framework called SIMaaS (simulation-as-a-service) [102]. We enhanced the SIMaaS architecture to account for various modes that SBOaaS has to operate in. In addition, we added a new scheduling policy based on the SBOaaS requirements. The architecture is composed of both design time and runtime components that we describe in this section.

3.4.1 Runtime Architecture

Figure 3.3 illustrates the key components of SBOaaS. **SIMaaS Manager (SM)** is at the core of the framework and is responsible for coordinating other components, handling user requests and decision-making. SM's pluggable architecture allows it to switch between various virtualization technologies and scheduling policies. The earlier framework was composed of deadline based scheduler where the number of simulation tasks to execute was known a priori. However, in the current work, the simulation count is not known a priori and additional constraints were introduced for synchronous and asynchronous modes that required relaxation of system level deadline constraint in favor of resource optimization based on the intermediate results. Thus, we introduced a greedy scheduling policy that leverages the intermediate results to maximize performance of the optimization algorithm and saturates the resources to minimize under utilization.

The simulation cloud deploys on a host cluster constructed by using the Docker[75] container virtualization technology. A Docker host can run multiple Docker containers, each representing a single computational node in the cloud system. Each simulation-based optimization task runs in a single container. The entire life cycle is managed by the **Container Manager (CM)** shown in Figure 3.3. CM supports various virtualization technologies such as KVM, however, due to its low startup and tear-down duration, we opted for Docker containers. The role of the CM includes management of hosts, execution, tear-down and deployment of the containers. The CM also maintains a registry for Docker images submitted at design time (explained in Section 3.4.2).

Another key component of SBOaaS is the **Result Aggregator (RA)** which is responsible for the collection of results from the simulation containers after they finish their tasks. It also performs result aggregation and informs the SM. The aggregator applies a message queue such that it does not get overloaded with simultaneously finishing simulation tasks. Based on the different aggregation requirements for SBOaaS compared to SIMaaS, we developed sync and async modes for the RA.

- **ASync Aggregator**. The asynchronous aggregator informs the SM as soon as the client aggregator logic aggregates the intermediate results based on the finised simulation task. The SM in turn replaces the old simulation instance and hence keeps the allocated resources 100% utilized.

- **Sync Aggregator**. Sync aggregator waits for all the simulation tasks to finish from the current cycle and invokes the client aggregation logic to obtain the intermediate result such that the next set of tasks can be executed. This helps in initializing the next cycle with the based result. However, this also results in resource under-utilization.

The final piece in the runtime architecture is the **Performance Monitor (PM)** which works with the CM to collect performance metrics from the host cluster and periodically informs the SM for decision making.

3.4.2 Design Time Architecture

The application designer interacts with the SBOaaS interface at design time to provide the configuration, executables and aggregation logic. The designer enters a list of configuration properties using a template that includes the execution command for the simulation task, the expected runtime input parameters, and desired resources among others. The designer also provides the simulation executable in the form of a container image which is uploaded by the system to the image registry and later deployed on the hosts by the CM during runtime. Please note that the first iteration of the simulation tasks incurs an additional deployment cost due to the image download time. This can be avoided by scheduling the simulation jobs *a priori*.

Another key role of the designer is to provide the aggregation logic using the SBOaaS aggregator template which is hooked to the Result Aggregator (RA). In this work, the aggregation logic is the optimization algorithm. However, this may vary from one use case to the other.

3.4.3 User Interaction Framework

The SBOaaS interface resides on a light-weight web framework to interact with the system designers, users or APIs and also to provide the result to the invoker. If the deadline is not immediate, user can provide the runtime parameters using web forms and collect the result from the download link returned by the simulation manager (SM) from the web

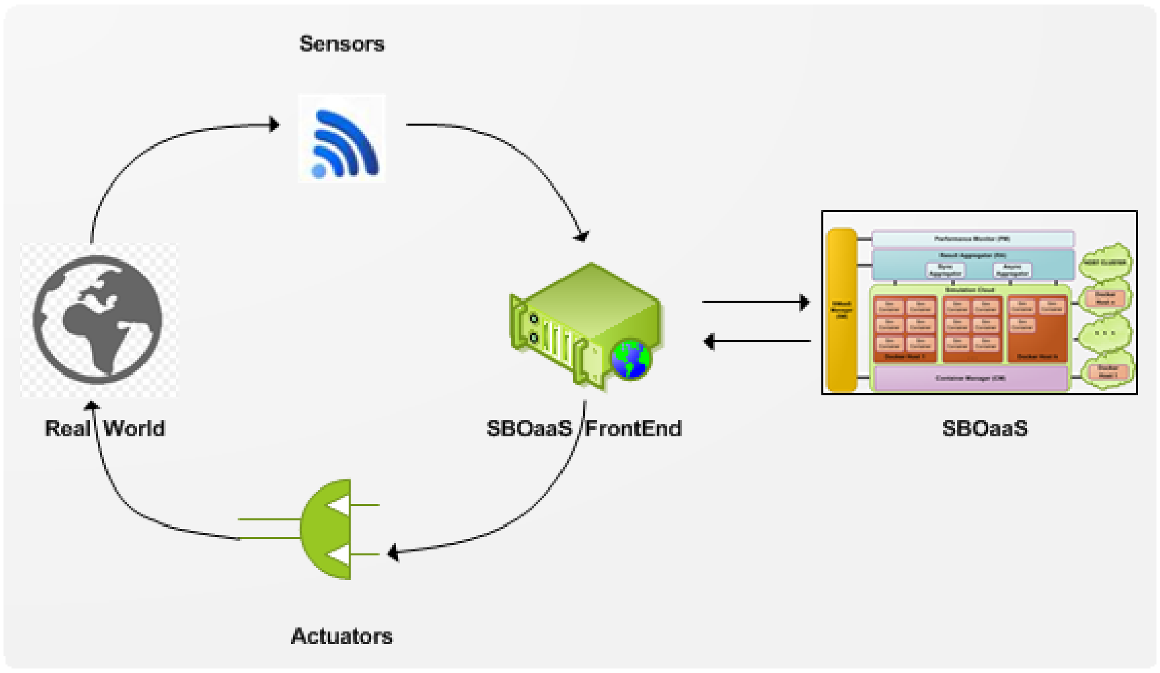


Figure 3.4: System Interaction

server.

In a typical system, the manual steps are eliminated with the use of APIs. Figure 3.4 depicts how the SBOaaS interacts with the real world and provides solutions to the optimization problems. Another server labeled as SBOaaS FrontEnd receives runtime parameters for the simulation based optimization in the form of aggregated sensor data. This FrontEnd invokes RESTful APIs from the SBOaaS interface to start a simulation job. Once the job is completed, the results are collected by the SBOaaS FrontEnd and actuation is performed based on the optimization results.

3.5 Evaluation

3.5.1 Online simulation-based optimization for dynamic traffic light control system

3.5.1.1 Environment

The simulation environment is defined according to the dynamic traffic light control scenario described in Section 3.2. To simulate the controlled traffic flow, we employ a simulation suite called SUMO [8] (short for "Simulation of Urban MObility"). SUMO is an open source, highly portable, microscopic road traffic simulation package designed to handle large road networks. SUMO also provides a Traffic Control Interface (TraCI) to let external controllers control the traffic. In our work, we use a Python script to control the simulation through TraCI and implement our control algorithm. The experiment environment is encapsulated into a Docker image in order to be distributed among the computing nodes through SBOaaS.

Our framework was deployed on NSF Chameleon cloud services, which is a cloud platform funded by National Science Foundation (NSF), providing such a large-scale platform to the research community allowing them to explore transformative concepts in deeply programmable cloud services, design, and core technologies. In the experiments, we created a distribution system with 8 computing nodes and 384 cores.

The input data are the map of the Vanderbilt University campus including all exogenously specified parameters (phase sequences and min-max intervals) and the corresponding vehicle flows in a morning scenario based on observations of road sensors. 9 intersections were selected to deploy the feedback controllers. In addition, we only consider two phases for each intersection to have dynamic intervals, which means there are two thresholds for each intersection that need to be optimized. Thus, for 9 intersections, the optimization problem dimension is 18. We consider variable domain $\{1, ..., 20\}$ and use the vehicle average speed to measure performance.

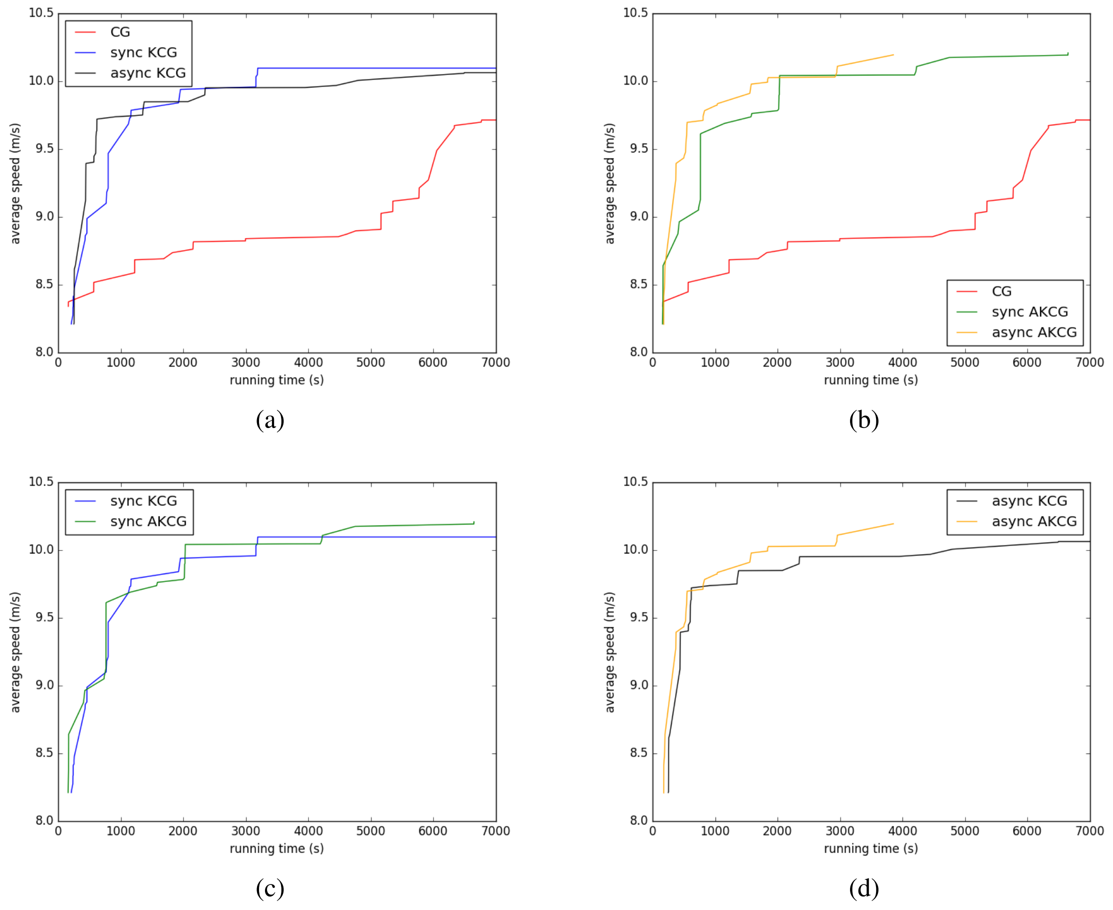


Figure 3.5: Comparison between coordinate greedy, synchronized/asynchronous K-coordinate greedy, as well as synchronized/asynchronous adaptive K-coordinate greedy decentralized solutions.

3.5.1.2 Experiment 1

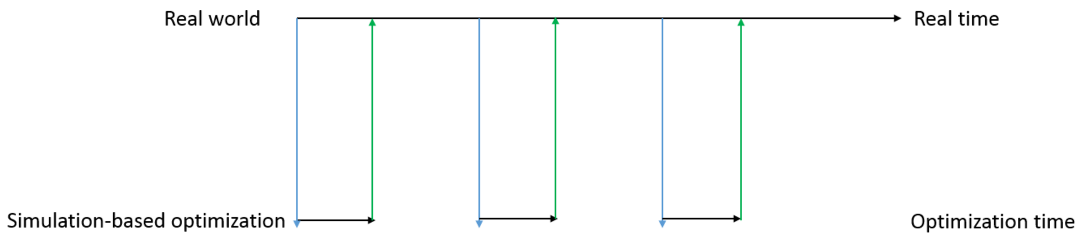
We first evaluate the performance of anytime optimization methods used in SBOaaS. The experiments ran until either the local optimum is found or the deadline (7000 seconds) is reached (*K*-coordinate greedy does not check convergence because there is no well-defined termination test).

3.5.1.3 Results

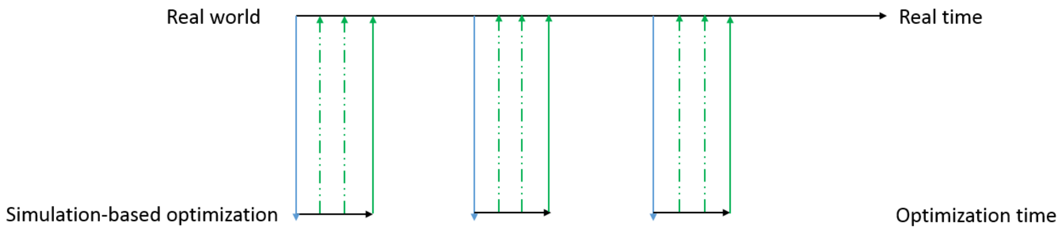
The experiment results can be seen in Figure 3.5, which shows simulation outcome (average vehicle speed) as a function of the running time of the optimization process. Figure 3.5a and Figure 3.5b, respectively, compare the coordinate greedy algorithm (Algorithm 2) with both K-coordinate greedy algorithm and adaptive K-coordinate greedy algorithm (Algorithm 3 and Algorithm 4). The optimization process is significantly accelerated by the variable-level parallelism. In general, both K-coordinate greedy and adaptive K-coordinate have the same rate of convergence. However, Figure 3.5c and Figure 3.5d indicate that K-coordinate greedy failed to converge within the deadline while adaptive K-coordinate greedy found the local optimum. The asynchronous algorithm has better performance, and a “smoother” curve than the synchronized one, which means better anytime response for returning sub-optimal solutions.

3.5.1.4 Experiment 2

Figure 3.6 illustrates the control processes of DDDAS with traditional simulation-based optimization and any simulation-based optimization. For the current observation, the former one gets and updates the optimum control parameters at the end of optimization process, while the latter one can continuously refresh the control parameters. We now compare both approaches. In this experiment, one simulation was run to simulate the morning scenario in real world. We also start the optimization service simultaneously with the initial road vehicle flow observation, and periodically updated the corresponding sub-optimal control parameters with real time line. We used asynchronous adaptive K-coordinate greedy algorithm and only consider the first optimization period, which is from 7:00 am to 7:30 am (asynchronous adaptive K-coordinate greedy converged within 30 minutes according to the last experiment). We considered several different periods for updating control parameters.



(a)



(b)

Figure 3.6: (a) DDDAS with traditional simulation-based optimization (b) DDDAS with anytime simulation-based optimization

3.5.1.5 Results

Period	Overall average speed (m/s)
baseline	8.703
1 sec	9.961
5 min	9.918
10 min	9.566

The experiment results can be seen in Figure 3.7, which shows the average instantaneous vehicle velocity in simulation area as a function of real time and the overall average speed is given by the table. The baseline is the situation that the "real world" run without control parameters updating, which is the behavior of DDDAS with traditional simulation-based optimization. In Figure 3.7, the instantaneous outcomes of the tested optimization methods do not show significant differences at the early stage. As the process of optimization gets better and better sub-optimal control parameters, the anytime optimization

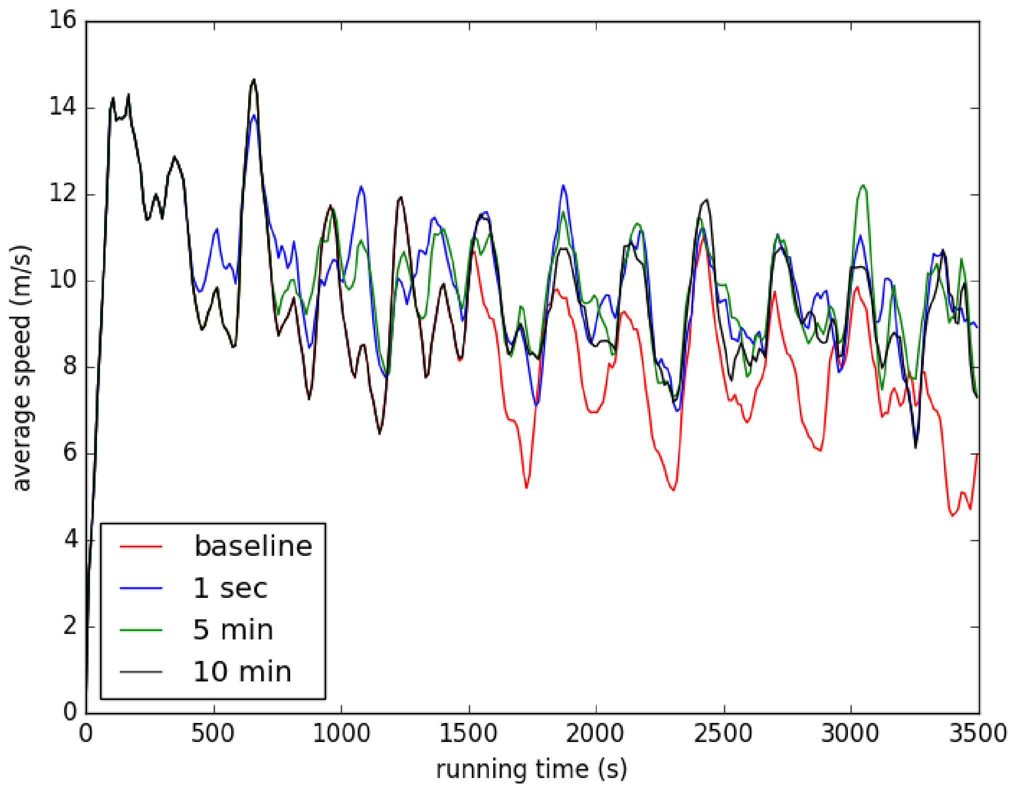


Figure 3.7: Experiment 2

services gradually improve the outcomes. From the overall performance, the optimization processes with shorter updating intervals gained better outcomes, but got less improvements. Limited by technologies and costs, a real world DDDAS usually will need to choose a suitable updating frequency by considering its marginal benefit.

3.5.2 System Evaluation

We measured the system metrics to validate its robustness and evaluated the overhead. The test bed was setup in accordance with the architecture shown in Figure 3.3. The SIMaaS Manager and the Result Aggregator were deployed on the same machine. Eight simulation hosts were added to the setup each having 48 cores. Two experiments were performed to assess the performance when the system is running in both synchronous and asynchronous modes. Each experiment was performed for 90 minute duration.

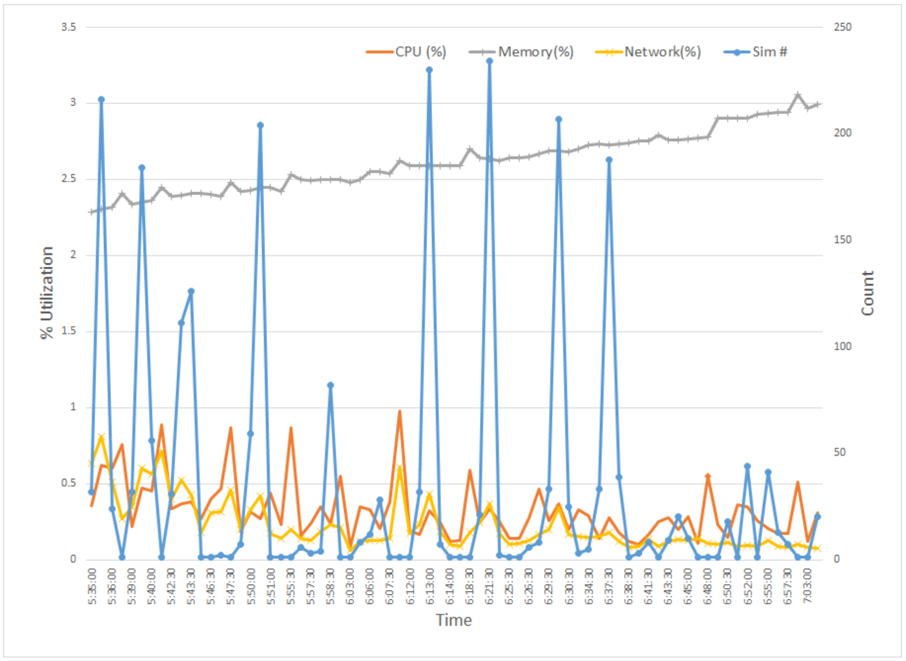
3.5.2.1 Results

The results of the experiments are illustrated in Figure 3.8. We observed that during the 90 minute duration, 2520 simulations were performed in synchronous mode and 11037 in asynchronous mode. The higher number of simulations for asynchronous mode is expected as the goal is to fully utilize the available servers. The figure also displays the utilization metrics of the management server. In both the modes, the CPU and network utilization is less than 1%. The memory utilization is around 3% for asynchronous mode and around 7% for synchronous mode. Even though we see an initial upward trend in memory utilization due to auditing of simulation tasks, it stabilizes towards the end of the experiment because of the cleanup operations running periodically to clear the old simulation containers. We also observe that the spikes in CPU and network usage is low when the simulation tasks are scheduled and when they finish. These results demonstrate that our architecture is robust with low overhead.

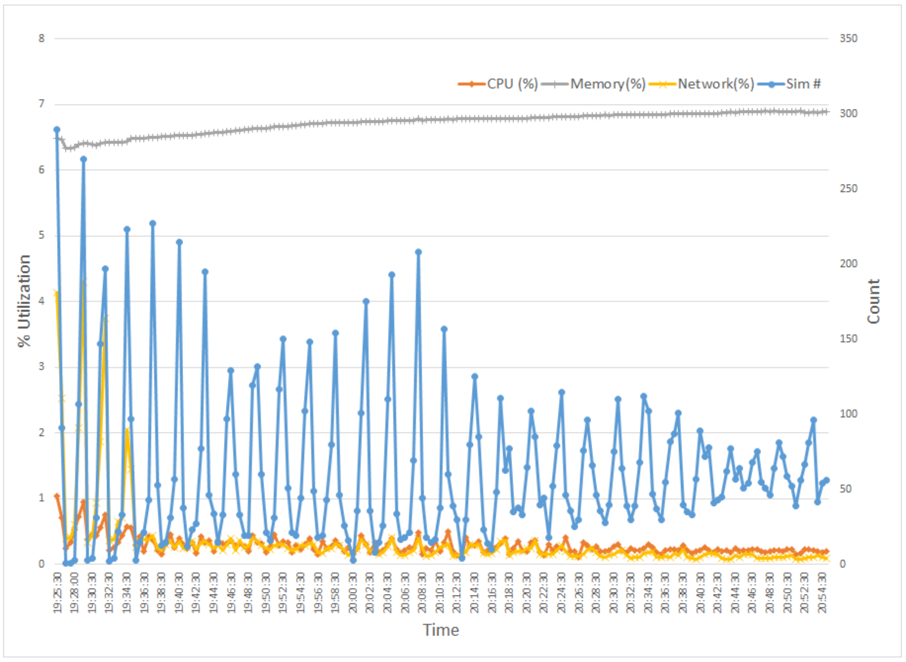
Figure 3.9 is the scatter plot for the simulation task execution times for both the modes. We see periodicity in the number of simulation tasks completing in the two modes. There are stragglers in the system which has higher impact on the synchronous mode compared to asynchronous mode as all the tasks of the next cycle have to wait for few stragglers to perform execution. In future, straggler management policies will be implemented which will significantly benefit the synchronous mode.

3.6 Conclusion

We presented a framework for simulation-based optimization as a service, which is a fundamental facility for DDDAS. We proposed a system architecture of our framework, and anytime optimization approaches including several coordinate decent method algorithms for solving simulation-based optimization problems in parallel. Then, we presented the dynamic traffic light control system as a case study. Finally, we evaluated both our anytime optimization algorithms and the online closed loop pattern.



(a) Sync Mode



(b) Async Mode

Figure 3.8: System Utilization vs Completed Simulation Count

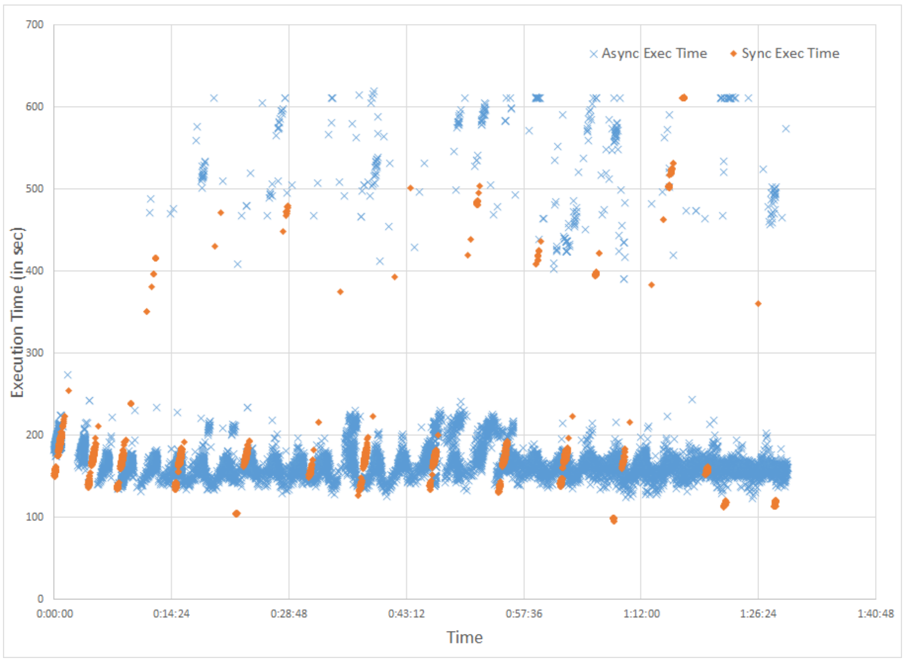


Figure 3.9: Execution Time

Chapter 4

Path Planning Game

4.1 Problem Overview

We investigate strategic interactions among path planning agents using a game theoretic path planning framework. Our focus is on economic tension between two important objectives: efficiency in the agents' achieving their goals, and safety in navigating towards these. We begin by developing a novel mathematical formulation for path planning that trades off these objectives, when behavior of other agents is fixed. We then use this formulation for approximating Nash equilibria in path planning games, as well as to develop a multi-agent cooperative path planning formulation. Through several case studies, we show that in a path planning game, safety is often significantly compromised compared to a cooperative solution.

4.2 Model

We describe the problem by first introducing the model of agents' motions, and then formulating the path planning game.

Consider a state space $\mathscr{X} = \mathbb{R}^n$. We represent an agent i by a polyhedron described by a collection of M_i hyperplanes: $P_i = \{a_{ij}^T x \leq b_{ij}, j \in \{0,...,M_i\}\}$. Each agent polyhedron P_i contains a point $r_i \in \mathscr{X}$ called the *reference* which rigidly attaches to the polyhedron such that the state of an agent can be determined by the position of its *reference*. We assume that agents move in discrete time, and a control input $u_{it} \in \mathscr{U}_{it} \subset \mathbb{R}^m$ applied to the ith agent at time t moves the agent from state $r_{i,t} \in \mathscr{X}$ at time t to state $r_{i,t+1} \in \mathscr{X}$ at time $t+1$ according to a linear stochastic dynamic model

$$r_{i,t+1} = A_i r_{it} + B_i u_{it} + \omega_i, \tag{4.1}$$

where $A_i \in \mathbb{R}^{n\times n}, B_i \in \mathbb{R}^{n\times m}$, and $\omega_i \sim \mathcal{N}(0, \Sigma_i)$ is the process noise for ith agent at time t following an n-dimension zero-mean Gaussian distribution with a covariance matrix Σ_i.

For each agent we are given its initial placement $r_0 \in \mathscr{X}$ (i.e., where the agent starts) and a goal $r_{goal} \in \mathscr{X}$ which the agent needs to reach. Let $r_{i,0:T} =< r_{i0}, ..., r_{iT} >$ be a state sequence of the (reference point of the) ith agent from time 0 to T and $u_{i,0:T} =< u_{i0}, ..., u_{iT} >$ be a corresponding control sequence. However, once the agent reaches its goal, it remains there deterministically, and has no effect on other agents. We aim to find the optimal control sequence for the ith agent in this stochastic motion model, with the following criteria in mind:

1. After applying the resulting control sequence, the expected terminal position of the ith agent is $r_{i,goal}$,
2. the upper bound of the probability that the ith agent collides with other agents should be minimized, and
3. the agent reaches the goal in as few time steps as possible.

For the moment, we allow no feedback from observed state to control; we relax this restriction below.

Path Planning Game: Given these models of individual agents, we define a *path planning game* by a collection of N agents, with each agent i's action space comprised of all possible control sequences, $\prod_{t=0}^{T} \mathscr{U}_{it}$. In this game, each agent aims to compute an optimal control sequence, given the behavior of others, trading off two objectives: efficiency, or the number of times steps it takes to reach the goal, and safety, or the probability of collision. To formalize, let T_i be the expected number of times steps to reach the goal (if no collision occurs), and G_i the *safety margin*, related to the upper bound on the probability of collision as discussed below. An agent i's objective is then

$$J_i(u_{i,0:T_{max}}, u_{-i,0:T_{max}}) = \lambda T_i + (1-\lambda)G_i, \tag{4.2}$$

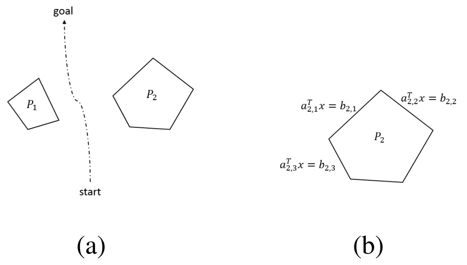


Figure 4.1: Single agent path planning with a point-like agent.

What makes this a game is that the safety G_i of an agent i depends on the paths taken by *all agents*, rather than i alone. For example, if two agents are moving towards one another, and directly towards their respective goals, the only way for one of them to avoid collision is to circumnavigate the other, taking a longer path towards the goal. Next, we describe how to define and compute T_i and G_i, and compute a *best response* for a given agent i, fixing behavior of all others.

4.3 Computing an Agent's Best Response

An important subproblem of computing a Nash equilibrium of a path planning game is to compute a best response of an arbitrary agent i when we fix the control policies of all others. We show that calculating agents' best responses in path planning games amounts to a single-agent path planning problem with motion uncertainty. Blackmore et al. [13] previously developed a probabilistic approach for computing a robust optimal path for a robot in the environment with a static obstacle and motion uncertainty via mathematical programming. However, in our context, where an agent trades off efficiency and safety, with stochastic *moving* obstacles (representing other agents), this prior approach is inadequate. In this section we develop a novel method for solving such problems.

4.3.1 Best Response for a Point-Like Agent

First, consider a simple path planning problem illustrated in Figure 4.1. In this problem, there is a set of static obstacles and an agent, represented by a point, aiming to find a collision-free minimum-time path from its initial placement to its goal position under motion uncertainty. Assume each obstacle has a given collision volume which can be represented by a polyhedron. To create a mathematical program for solving this problem, two factors need to be taken into account: goal position constraints and collision avoidance constraints.

Formally, let r_t denote the position of an agent at time t with its initial placement r_0 and the goal position r_{goal}. Suppose that the motion dynamics of the agent follows (4.1) (from which, we remove the index i, since there is only one agent). Assume there are K obstacles represented by polyhedra $P_n, n = 1, ..., K$, with $P_n = \{x | a_{np}^T x \leq b_{np}, p = 1, \ldots, E_n\}$, where E_n is the number of hyperplanes representing the polyhedron P_n. As before, let T denote the planning horizon (so that the goal must be reached by time T; we assume the horizon is long enough that the goal can be successfully reached even with the obstacles).

Efficiency and Reachability: Let $\{d_0, ..., d_T\}$ denote a collection of binary indicators which indicate whether the agent has reached its goal, i.e., $d_t = 1$ iff $r_t = r_{goal}$. Then, with a large positive number M, the constraints

$$\forall t, ||r_t - r_{goal}|| \leq M(1 - d_t) \tag{4.3}$$

$$\sum_{t=0}^{T} d_t = 1 \tag{4.4}$$

make sure that the agent will reach to its goal position sooner or later (and we assume that there exists a feasible solution). Moreover, the number of time steps to reach its goal position can be represented by

$$T = \sum_{t=0}^{T} t \cdot d_t \tag{4.5}$$

which is one of our objectives (corresponding to T_i, for an agent i above). Since r_t is a random variable, this constrain is stochastic. We approximate it by a deterministic constraint, replacing the position of the agent r_t with its expected position $\bar{r}_t$ in Constraint (4.3).

Collision Avoidance: Let A denote the event that the agent has a collision, and let $A(n,t), n \in \{1, ..., K\}$ denote the event that the agent collides with the nth obstacle at time step t. We wish to minimize the probability of a collision, $Pr(A)$, or minimize G such that

$$Pr(A) \leq G. \tag{4.6}$$

The agent has a collision if the agent collides with any of obstacles at any time steps, which is the event

$$A = \bigvee_{t=0}^{T} \bigvee_{n=1}^{K} A(n,t) \tag{4.7}$$

Then, by the union bound

$$Pr(A) = \Pr\left(\bigvee_{t=0}^{T} \bigvee_{n=1}^{K} A(n,t)\right) \leq \sum_{t=0}^{T} \sum_{n=1}^{K} \Pr(A(n,t)) \leq G \tag{4.8}$$

$$\Leftarrow [\forall n,t, Pr(A(n,t)) \leq g(n,t)] \wedge [\sum_{t=0}^{T} \sum_{n=1}^{K} g(n,t) = G], \tag{4.9}$$

where $g(\cdot)$ is *risk allocation* which indicates how the risks are distributed among obstacles and time steps. Next, we consider the event that the agent collides with an obstacle at time step t, which means that the position of the agent is inside the corresponding polyhedron. Thus, collision with the nth obstacle can be described by

$$A(n,t) : \bigwedge_{p=1}^{E_n} a_{np}^T \cdot r_t \leq b_{np} \tag{4.10}$$

Since the condition (4.10) including r_t is also stochastic, to convert it into a deterministic one, we consider its probabilistic measure, $\Pr\{A(n,t)\}$. Following (4.6), our constraints then become

$$\Pr\left\{\bigwedge_{p=1}^{E_n} a_{np}^T \cdot r_t \leq b_{np}\right\} \leq g(n,t). \tag{4.11}$$

Since a polyhedron is convex, a sufficient condition is,

$$\bigvee_{p=1}^{E_n} \Pr(a_{np}^T \cdot r_t \leq b_{np}) \leq g(n,t). \tag{4.12}$$

Based on the approach by Blackmore et al. [13], expression (4.11) can be further simplified using the linear approximation of the upper bound on the probability of collision. First, consider r_t, the position of agent at time step t given its initial placement r_0 and the control sequence $u_{0:t}$, which is a random variable following a Gaussian distribution, $r_t \sim N(\bar{r}_t, \Sigma_t)$, where

$$\bar{r}_t = \sum_{k=0}^{t-1} A^{t-k-1} B u_k + A^t r_0 \tag{4.13}$$

and

$$\Sigma_t = \sum_{k=0}^{t-1} A^{t-k-1} \Sigma (A^T)^{t-k-1}. \tag{4.14}$$

For a single Gaussian random variable $X \sim N(\mu, \sigma^2)$, we can take the inverse Gaussian distribution function at both sides of $\Pr(X < 0) \leq \delta$ and get $u \geq \sqrt{2}\sigma erf^{-1}(1-2\delta)$. Similarly, from $r_t \sim N(\bar{r}_t, \Sigma_t)$, we can get $(a_{np}^T r_t - b_{np}) \sim N(a_{np}^T \bar{r}_t - b_{np}, a_{np}^T \Sigma_t a_{np})$. Then, we take the inverse Gaussian distribution function at both sides of (4.12), and

$$\bigvee_{p=1}^{E_n} a_{np}^T r_t - b_{np} \geq e(n,t) \tag{4.15}$$

where $e(n,t) = \sqrt{2 a_{np}^T \Sigma_t a_{np}} \cdot erf^{-1}(1 - 2g(n,t))$ and $erf(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$. We call this the *safety margin*, because it expands the margin of obstacles and shrinks the feasible planning domain in order to consider motion uncertainty. Because the motion of the agent after it reaches its goal has no further effect, we add the term $M \sum_{k=0}^{t} d_k$ to these constraints where M is a large positive number.

Define $s(n,t) = erf^{-1}(1-2g(n,t))$. Since erf^{-1} is strictly monotonically increasing, we can minimize $\sum_{t=0}^{T}\sum_{n=1}^{K} g(n,t)$ by minimizing

$$G = -\sum_{t=0}^{T}\sum_{n=1}^{K} s(n,t). \tag{4.16}$$

This is the *safety* portion of an agent's objective (G_i for an agent i above).

A Path Planning Mathematical Program: Our goal is to minimize $J = \lambda T + (1-\alpha)G$, balancing efficiency and safety using an exogenously specified parameter λ. Combining this objective with the goal and collision avoidance constraints described above, we obtain the following mathematical program for single-agent path planning:

MP1:

$$\min_{u,s,d} \lambda T(d) + (1-\lambda)G(s) \tag{4.17}$$

s.t.

$$\forall t, u_t \in \mathscr{U}_t \tag{4.18}$$

$$\forall t, \bar{r}_t = \sum_{k=0}^{t-1} A^{t-k-1}Bu_k + A^t r_0 \tag{4.19}$$

$$\forall t, ||\bar{r}_t - r_{goal}||_1 \leq M \cdot (1-d_t) \tag{4.20}$$

$$\forall t, d_t \in \{0,1\} \tag{4.21}$$

$$\sum_{t=0}^{T} d_t = 1 \tag{4.22}$$

$$\forall t \forall n, \bigvee_{p=1}^{E_n} a_{n,p}^T \bar{r}_t > b_{np} + e(n,t) - M\sum_{k=0}^{t} d_k \tag{4.23}$$

$$\forall t, e(n,t) = s(n,t)\sqrt{a_{np}^T \Sigma_t a_{np}} \tag{4.24}$$

$$\forall t, \Sigma_t = \sum_{k=0}^{t-1} A^{t-k-1}\Sigma(A^T)^{t-k-1} \tag{4.25}$$

$$\forall t \forall n, 0 \leq s(n,t) \leq M' \tag{4.26}$$

One residual concern is that if an agent cannot possibly collide with an nth obstacle at time step t (i.e., if $g(n,t) = 0$), $s(n,t)$ can become unbounded. To address this, we add Constraint (4.26) which imposes an upper bound M' on $s(\cdot)$, where M' is an appropriate positive number so that $erf(M') \simeq 1$.

Since MP1 is a disjunctive linear program which can be solved by an off-the-shelf linear programming solver. A solution $< u, s(\cdot), d >$ found by MP1 with $d_{T_0} = 1$ means that the agent can reach to its goal position in T_0 time steps with the probability of collision at most $\sum_{t=0}^{T_0} \sum_{n=1}^{K} \frac{1-erf(s(n,t))}{2}$ by applying the control sequence $u_{0:T_0}$.

4.3.2 Generalization: Feedback Control

Above we considered *open loop* path planning where the control sequence is deterministic and fixed a priori. We now extend our approach to *closed loop (feedback)* control, following the ideas in Geibel and Wysotzki [35] and Oldewurtel et al. [83].

Assume we have a nominal control sequence $\overline{u}_{0:T}$. Then, the feedback control sequence can be obtained by integrating the nominal control sequence and the feedback gain:

$$u_t = \overline{u}_t + K(x_t - \overline{x}_t), \tag{4.27}$$

where x_t is the observed and $\overline{x}_t$ the predicted position, and K is an exogenous parameter which determines the importance of the error feedback term $(x_t - \overline{x}_t)$. In this approach, $\overline{u}_t$ is computed using the MP1 offline, and the actual control sequence is then generated at runtime by applying (4.27). As a consequence, the Constraints (4.25) above become

$$\Sigma_t = \sum_{k=0}^{t-1} (A+BK)^{t-k-1} \Sigma[(A+BK)^T]^{t-k-1}. \tag{4.28}$$

Notice that when there is no error feedback ($K = 0$) this becomes equivalent to open loop control.

4.3.3 Collision Avoidance for Polyhedral Agents

Having considered the problem for point-like agents, and then generalizing the approach to consider error feedback, we now generalize the collision avoidance constraints to polyhedral agents.

Consider states of the agent and the nth obstacle, both represented by polyhedra P_t and P_n, respectively. The position of the agents' reference is r_t. Since the reference point rigidly attaches to the agent, let $C = \{x - r_t | x \in P_t\}$ denote the relative region of the agent to its time-dependent reference. When the agent collides with the nth obstacle at time t, we know that $\exists x \in P_t \cap P_n$ (i.e., the intersection of these time-dependent polyhedra is non-empty). Thus, from the point view of the agent, the set of positions of its reference causing collision with the nth obstacle can be represented by $K_n = \{x - c | x \in P_n, c \in C\} = -C \oplus P_n$, where $\oplus$ is the Minkowski addition. Since both C and P_n are polyhedra, K_n is a polyhedron and can be represented by a set of hyperplanes: $K_n = \{x | a_{np}^T x \leq b_{np}, p = 0, ..., E_n\}$, where E_n the number of hyperplanes of K_n. The agent collides with the nth obstacle at time step t if the position of its reference is in K_n, that is, when

$$r_t \in K_n \Leftrightarrow \bigwedge_{p=1}^{E_n} a_{np}^T r_t \leq b_{np}. \tag{4.29}$$

Comparing (4.29) with (4.10), we can see that the problem with polyhedral agents can also be solved via the mathematical program above, if we treat the agent as its reference point, and assign the collision volume K_n to each obstacle.

4.3.4 Best Response Solver

Our final challenge is to consider the actual best response problem of an arbitrary agent in the path planning game, where all other agents are *moving* (rather than static) obstacles with known stochastic motion policies. We now address this problem, obtaining the final mathematical program for computing a single-agent best response.

Let i denote the agent for whom we are computing a best response, with $-i = \{1, ..., i-1, i+1, ..., N\}$ the set of all others. Let i be represented by a polyhedron P_{it} with reference r_{it} and let $j \in -i$ be represented by P_{jt} with reference r_{jt}. Let C_i denotes the relative region of i to its reference, while C_j denotes the relative region of $j \in -i$ to its reference. Suppose that j reaches its goal position by time step T_j with the corresponding known control sequences $u_{j,0:T_j}$. Then, for each j and t, $K_{ijt} = -C_i \oplus P_{jt}$ is a polyhedron with $K_{ijt} = \{x | a_{ijp}^T x \leq b_{ijtp}, p \in \{0, ..., E_{ij}\}\}$ where E_{ij} is the number of hyperplanes related to the shapes of C_i and C_j.

Now we formalize how the control sequence $u_{j,0:T_j}$ of each agent j affects P_{jt} so that we can determine K_{ijt}. From motion dynamics of i and j,

$$r_{it} = \sum_{k=0}^{t-1} A_i^{t-k-1} B_i u_{ik} + A_i^t r_{i0} + \omega_{it} \tag{4.30}$$

$$\forall j,\ r_{jt} = \sum_{k=0}^{t-1} A_j^{t-k-1} B_j u_{jk} + A_j^t r_{j0} + \omega_{jt} \tag{4.31}$$

From the perspective of agent i, the motion of agent j can be treated as deterministic if we "migrate" motion uncertainty from j to i so that

$$\forall j, r'_{ijt} = \sum_{k=0}^{t-1} A_i^{t-k-1} B_i u_{ik} + A_i^t r_{i0} + \omega_{it} - \omega_{jt}$$

$$\forall j, r'_{jt} = \sum_{k=0}^{t-1} A_j^{t-k-1} B_j u_{jk} + A_j^t r_{j0}. \tag{4.32}$$

For each j, let $\omega'_{ijt} = (\omega_{it} - \omega_{jt}) \sim N(0, \Sigma_{it} + \Sigma_{jt})$ denote the relative motion uncertainty of i to j at time t. Let

$$\forall j, \Delta r'_{jt} = \sum_{k=0}^{t-1} A_j^{t-k-1} B_j u_{jk} + A_j^t r_{j0} - r_{j0} \tag{4.33}$$

denote the position shift of agent j at time step t determined by its control sequence $u_{j,0:T_j}$. Then, we obtain the position of K_{ijt} by shifting K_{ij0} by Δr_{jt}. Since $K_{ijt} = \{x | a_{ijp}^T x \leq b_{ijtp}\}$, we obtain

$$b_{ijtp} = b_{ij0p} + a_{ijp}^T \cdot \Delta r'_{jt}. \tag{4.34}$$

Consequently, we obtain the following mathematical program for i's best response:

MP2:

$$\min_{u,s_i(\cdot),d} J_i = \lambda T_i + (1-\lambda) G_i \tag{4.35}$$

s.t.

$$\forall t, u_{it} \in \mathscr{U}_{it} \tag{4.36}$$

$$\forall t, \bar{r}_{it} = \sum_{k=0}^{t-1} A_i^{t-k-1} B_i u_{ik} + A_i^t r_{i0} \tag{4.37}$$

$$\forall t, ||\bar{r}_{it} - r_{i,goal}||_1 \le M \cdot (1 - d_{it}) \tag{4.38}$$

$$\forall t, d_{it} \in \{0,1\} \tag{4.39}$$

$$\sum_{t=0}^{T} d_{it} = 1 \tag{4.40}$$

$$\forall j \forall t = 0,...,T_j,$$

$$\bigvee_{p=1}^{E_{ij}} a_{i,j,p}^T \bar{r}_{it} > b_{ij0p} + a_{ijp} \cdot \Delta r'_{jt} + e_{ijt}$$

$$-M \sum_{k=0}^{t} d_{ik} \tag{4.41}$$

$$\forall i \forall t, \Sigma_{it} = \sum_{k=0}^{t-1} (A_i + K_i B_i)^{t-k-1} \Sigma_i [(A_i + K_i B_i)^T]^{t-k-1} \tag{4.42}$$

$$\forall t \forall j, \Delta r'_{jt} = \sum_{k=0}^{t-1} A_j^{t-k-1} B_j u_{jk} + A_j^t r_{j0} - r_{j0} \tag{4.43}$$

$$\forall j, e_{ijt} = \sqrt{a_{ijp}^T (\Sigma_{it} + \Sigma_{jt}) a_{ijp}} \cdot s_i(j,t) \tag{4.44}$$

$$\forall t \forall n, 0 \le s_i(n,t) \le M' \tag{4.45}$$

Notice that the constraints (4.41) are effective only for $t = 0,...,T_i$, and i is not affected by any j who reached its goal.

4.4 Finding Equilibria in Path Planning Games

Armed with the best response solvers for each agent i in a path planning game, our goal is to approximate a Nash equilibrium in the resulting game. We do so by applying *best response dynamics* which, if it converges (which it does in our experiments), yields a Nash equilibrium.

Best response dynamics is an asynchronous iterative algorithm in which a single agent i is chosen in each iteration, and we maximize i's utility (i.e., compute its best response) fixing control strategies for all other agents. Best response of an agent i can be calculated as discussed above. Then, Nash equilibrium can be yielded by the best response dynamic (algorithm 5),

Algorithm 5 Best Response Dynamics

```
1:  **input** initial action, $u^{(0)}_{1,0:T}, ..., u^{(0)}_{N,0:T}$
2:  **output** Nash equilibrium $u^{*}_{1,0:T}, ..., u^{*}_{N,0:T}$
3:  Set $k \leftarrow 0$
4:  **repeat**
5:      **for** $i \leftarrow 1, ..., N$ **do**
6:          $u^{(k+1)}_{i,0:T} = \arg\min_{u_{i,0:T}} J_i(u_{i,0:T}, u^{(k)}_{N,0:T})$
7:      **end for**
8:      Set $k \leftarrow k+1$
9:  **until** $\forall i, u^{(k)}_{i,0:T} = u^{(k-1)}_{i,0:T}$
10: **return** $u^{(k)}_{i,0:T}$
```

4.5 Optimal Multi-Agent Path Planning

We now extend the single-agent best response problem to compute an optimal multi-agent path plan. In this case, the control sequences $u_{i,0:T_i}$ of all agents are unknown a priori (as they are being computed jointly). Compared to calculating an agents' best response, we replace the objective of the current agent with the sum of all agents' objectives, i.e., the new objective is $J = \sum_i J_i$, where J_i is the objective of agent i. Moreover, we add constraints analogous to MP2 to make sure that the collision avoidance conditions hold from the perspective of *every agent simultaneously*. We thus obtain the following mathematical program:

MP3:

$$\min_{u,s,d} J = \sum_{i=1}^{N} J_i \tag{4.46}$$

s.t.

$$\forall i,t, u_{it} \in \mathscr{U}_{it} \tag{4.47}$$

$$\forall i,t, \bar{r}_{it} = \sum_{k=0}^{t-1} A_i^{t-k-1} B_i u_{ik} + A_i^t r_{i0} \tag{4.48}$$

$$\forall i,t, ||\bar{r}_{it} - r_{i,goal}||_1 \leq M \cdot (1 - d_{it}) \tag{4.49}$$

$$\forall i,t, d_{it} \in \{0,1\} \tag{4.50}$$

$$\forall i \sum_{t=0}^{T_{max}} d_{it} = 1 \tag{4.51}$$

$$\begin{aligned} \forall i,t,-i, \bigvee_{p=1}^{E_{i,-i}} a_{i,-i,p}^T \bar{r}_{it} > b_{i,-i,0,p} + a_{i,-i,p} \cdot \Delta r'_{-i,t} \\ + e_{i,-i,t} - M \sum_{k=0}^{t} (d_{ik} + d_{-i,k}) \end{aligned} \tag{4.52}$$

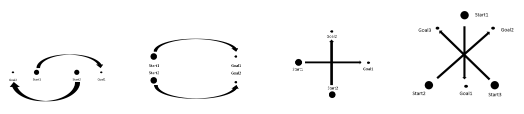
$$\forall i \forall t, \Sigma_{it} = \sum_{k=0}^{t-1} (A_i + K_i B_i)^{t-k-1} \Sigma_i [(A_i + K_i B_i)^T]^{t-k-1} \tag{4.53}$$

$$\forall i,t, \Delta r'_{it} = \sum_{k=0}^{t-1} A_i^{t-k-1} B_i u_{i,k} + A_i^t r_{i0} - r_{i0} \tag{4.54}$$

$$\forall i,t \forall -i, e_{i,-i,t} = \sqrt{a_{i,-i,p}^T (\Sigma_{it} + \Sigma_{-i,t}) a_{i,-i,p}} \cdot s_i(t,j)$$

$$\forall i \forall t \forall n, 0 \leq s_i(n,t) \leq M' \tag{4.55}$$

The term $-M \sum_{k=0}^{t} (d_{ik} + d_{-i,k})$ in Constraints (4.52) means that an agent will not be affected by other agents who have reached their goal position by time step t, and, conversely, it will not affect the final solution once it reaches its goal position.



(a) (b) (c) (d)

Figure 4.2: Experiment scenarios.

Armed with the techniques for computing both Nash equilibria in path planning games, as well as a socially optimal solution of the corresponding "cooperative" multi-agent planning scenario, we now consider several case studies to understand the impact of self-interested behavior. Specifically, we consider the following 2D scenarios:

- **2 agents with opposing goal positions (Figure 4.2a):** the goal position of each agent is behind the initial placement of the other. In this scenario, the first agent moves from starting coordinate position $(10,50)$ to goal at position $(95,50)$, and the second agent moves from $(90,50)$ to $(5,10)$.

- **2 agents moving in parallel (Figure 4.2b):** the initial and goal positions of both agents are near one another. In this scenario, the first agent moves from $(10,70)$ to $(95,70)$ and the second agent moves from $(10,35)$ to $(95,35)$.

- **Intersection with 2 agents (Figure 4.2c):** one agent moves from the bottom to the top of the 2D grid, and the other moves from left to right. In this scenario the first agent moves from $(10,50)$ to $(90,50)$ and the second agent moves from $(50,10)$ to $(50,90)$.

- **Intersection with 3 agents (Figure 4.2d)**: one agent starts at the top of a 2D grid and moves down, while the other two start at southeast and southwest, and move northwest and southeast, respectively. In this scenario the first agent moves from $(50,90)$ to $(50,5)$, the second agent moves from $(85,30)$ to $(11,73)$, and the third agent moves from $(14,29)$ to $(90,73)$.

In each experiment, each agent is represented by a square with each side of length 15 and parallel to either the x or the y axis. The control inputs are 2D velocity vectors and the maximum velocity of agents in both x and y direction is 10 (thus, $A = B = I$ in agents' motion dynamic). Agents' motion is distorted by a Gaussian distribution with the covariance matrix $1.9I$. For each scenario we consider solutions with and without feedback control, where the feedback gain for the latter was chosen to be $K = 0.5$. Throughout, we assume that all players are equally concerned about safety vs. efficiency; formally, all players share the same parameter λ.

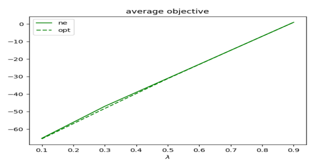
The results are shown in Figures 4.3-4.10. In each figure, the horizontal axis is the λ value which represents the importance of safety for both agents, where lower values of λ imply that safety is *more* important. The left plots show the objective value, where lower is better. The middle plots give the time to goal, where lower is, again, better. The right plots show safety margin, where again lower is better. We present average quantities over all agents; the qualitative observations are similar if we consider these at individual agent level.

The first observation is that the difference between socially optimal and equilibrium objective values appears small ((a) plots in Figures 4.3-4.10). It is therefore tempting to conclude that equilibrium behavior is similar to socially optimal, but it turns out that this is not the case: in particular, it turns out that the trade-off between efficiency and safety made by the agents in equilibrium is very different from optimal.

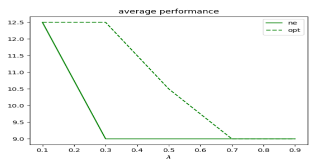
Considering next the (b) and (c) columns of the figures, we can observe that systematically performance improves, while safety is often significantly compromised, in equilibrium as compared to a social optimum. The difference is particularly dramatic in the first two scenarios, when the agents are in direct conflict in their quest to reach their respective goals. The gap between optimal and equilibrium safety in the other scenarios tends to be larger for relatively high values of λ.

Another general observation we can make is that often the solutions with a feedback controller are closer to optimal, particularly from the perspective of safety. The exceptions involve the intersection scenarios, where the gap is larger for higher values of λ in the feedback controller solution than with the open-loop controller. However, even in these scenarios, the feedback controller yields solutions closer to socially optimal for most values of λ. This is not surprising: since all agents are concerned about safety, they are more able to dynamically adjust to avoid collisions when some feedback about state is available.

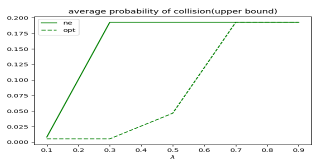
To understand why safety is systematically compromised, consider a single agent's incentive. Even though an agent is interested in reaching the goal safely, it does not account for the fact that being involved in a crash *also crashes the other agent*. Thus, in equilibrium safety is compromised relative to social optimum, as agents fail to capture the externalities associated with crashes.



(a)

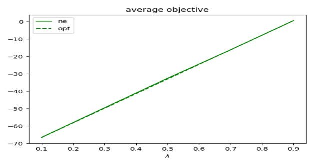


(b)

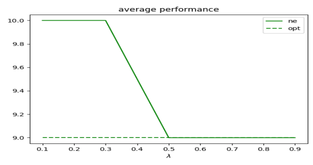


(c)

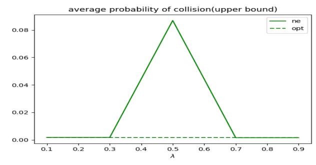
Figure 4.3: Opposing goal positions without the feedback gain($K = 0$).



(a)



(b)

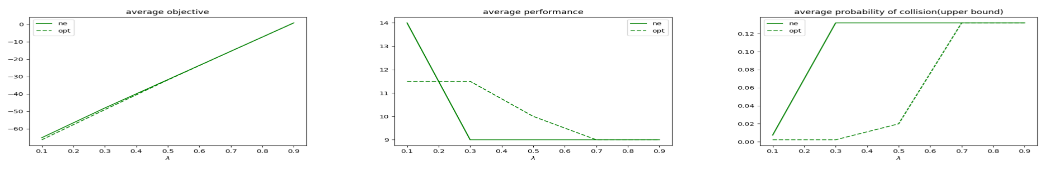


(c)

Figure 4.4: Opposing goal positions with the feedback gain($K = 0.5$).

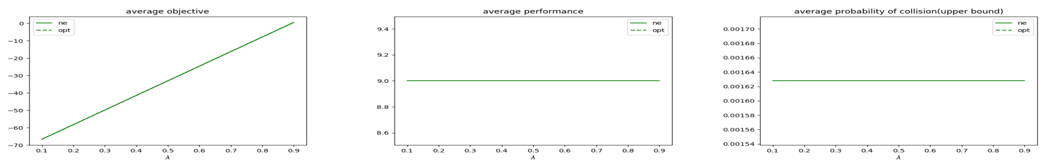
4.7 Conclusion

We study path planning games, which are strategic interactions among path planning agents who trade off efficiency in reaching their goals and safety (or collision avoidance). We construct mathematical programs for computing the best response of an arbitrary agent, fixing the policy of others, and a mathematical program for computing a socially optimal multi-agent path plan. The agents best response computation is then used as a part of asynchronous best response dynamics to compute a Nash equilibrium. Our experiments demonstrate that the outcomes of path planning games systematically compromise safety compared to socially optimal multi-agent path plans.



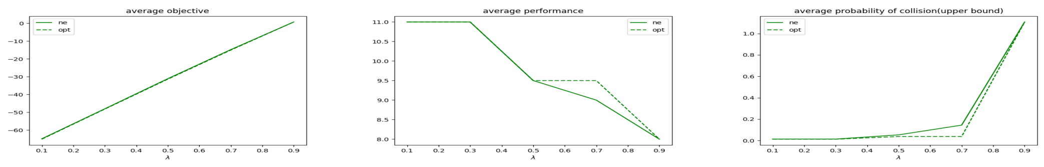
(d) (e) (f)

Figure 4.5: Moving in parallel without the feedback gain($K = 0$).



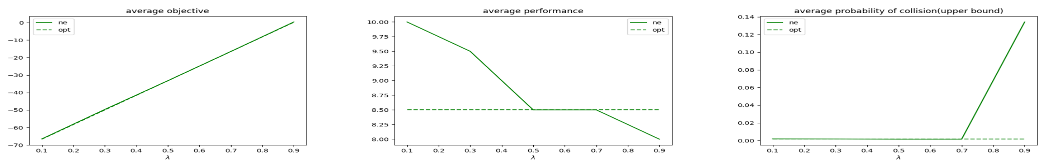
(g) (h) (i)

Figure 4.6: Moving in parallel with the feedback gain($K = 0.5$).



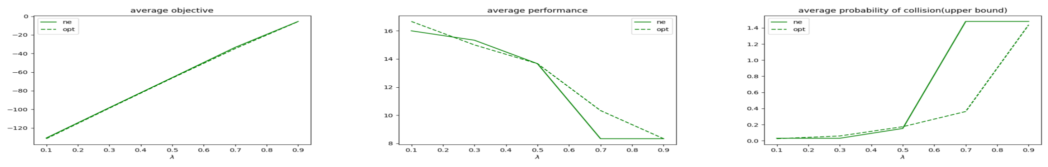
(a) (b) (c)

Figure 4.7: Intersection without the feedback gain($K = 0$, 2 agents).



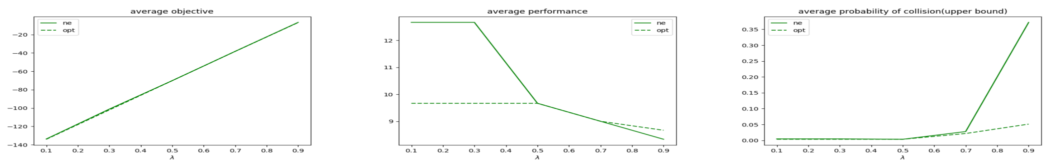
(a) (b) (c)

Figure 4.8: Intersection with the feedback gain($K = 0.5$, 2 agents).



(a) (b) (c)

Figure 4.9: Intersection without the feedback gain($K = 0$, 3 players).



(a) (b) (c)

Figure 4.10: Intersection with the feedback gain($K = 0.5$, 3 players).

Chapter 5

Robust Medical Imaging Recognition

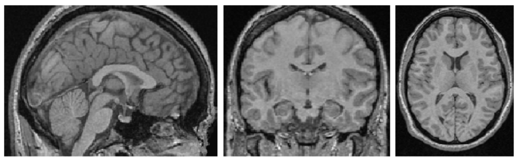
5.1 Problem Overview

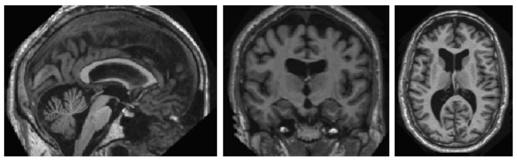
We present the first investigation of vulnerabilities of regression-based prediction in medical image processing to adversarial example attacks. Specifically, our problem setting involves predicting age of a subject based on their 3D MRI brain image, with malicious perturbations artificially injected directly into the digital images. Since prior adversarial example research is focused on classification or segmentation tasks, our first contribution is to adapt state-of-the-art methods for generating adversarial examples with l_0, l_2, and l_∞ constraints on the magnitude of the perturbation to our setting. Our second contribution is a method for generating *universal* adversarial perturbations for our domain—that is, a single perturbations (for each norm) that is effective on a large batch of images; this is entirely novel in the context of medical imaging. Our third contribution is to experimentally demonstrate that adversarial examples—both image-specific, and universal—are indeed extremely effective, significantly reducing prediction effectiveness of deep learning for age prediction. The observation of the effectiveness of universal perturbations in this setting is particularly powerful: it implies that a single malfunction in MRI equipment (inadvertent, or adversarial) can have a significant impact.

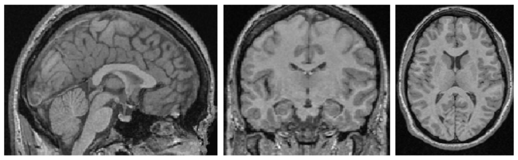
Given vulnerability of deep learning for medical imaging, it is natural to wonder whether one can effectively mitigate this issue. We explore one approach which has not previously been considered for this: augmenting deep learning models with volumetric features obtained through traditional multi-atlas segmentation techniques, where each feature corresponds to the volume of a brain region (for a total of 132 features). Such contextual information has previously been shown effective in improving prediction in non-adversarial settings [10, 57, 118], and is by construction relatively insensitive to small perturbations.

Our forth contribution is to demonstrate experimentally that, indeed, adding contextual features to deep learning significantly mitigates its vulnerability to adversarial perturbations, whether they are designed for each image independently, or universally crafted for batches of images.

To illustrate the effect of adversarial noise, consider Figure 5.1, where age is predicted using a conventional deep neural network. For this figure, we identify one sample with predicted age 19 and another sample with predicted age 80 (Figures 5.1a and 5.1b, respectively); we note that predictions on unperturbed data are extremely accurate (root mean squared error, RMSE < 5.13 years). Comparing the brain images of a 19 and 80 year-old, we can readily see clear differences between them. Next, we add low-magnitude random noise (l_∞ of noise is 0.002, where pixel values are normalized between 0 and 1) to the first (19-year-old) sample (Figure 5.1c), showing that the deep neural network is robust to such random perturbations. Figure 5.1d contrasts this with an adversarial perturbation *of the same magnitude*, but which causes the neural network to predict that the subject's age is 80!


(a) Age prediction: 19


(b) Age prediction: 80


(c) Age prediction: 19

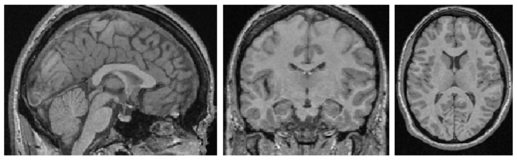

(d) Age prediction: 80

Figure 5.1: The illustration of the effect of adversarial attack. (a) Sample 1 (19 year-old). (b) Sample 2 (80 year-old). (c) Sample 1 with random noise. (d) Sample 1 with adversarial perturbation. The difference among (a), (c) and (d) appears imperceptible to human eye.

5.2 Methods

First, we describe the approaches we used to generate adversarial perturbations for a single image. Subsequently, we present our approach that targets a batch of images with a single perturbation. All our code is publicly available at https://github.com/yvorobey/adversarialMI.

5.2.1 Generating Adversarial Perturbations for a Single Image

Let $F(x)$ be the function computed by the deep neural network to predict age for an arbitrary input image x. Consider a fixed image x_0. Our goal is to generate a small (imperceptible) perturbation, Δx, to add to the original image x_0, so as to maximize or minimize predicted age. As described earlier, we use l_∞, l_2, and l_0 norms to quantify the magnitude of the introduced perturbation. In all cases, if we wish to maximize predicted age, the goal is to solve the following problem:

$$\begin{aligned} &\text{maximize}_{\Delta x} \quad G(\Delta x) = F(x_0 + \Delta x) \\ &\text{subject to:} \quad ||\Delta x||_p \leq \varepsilon, \quad x_0 + \Delta x \in [0,1]^n \end{aligned} \tag{5.1}$$

where $||\cdot||_p$ corresponds to the one of the above norms ($p = \infty, 2$, and 0, respectively), $F(x + \Delta x)$ is the predicted age for the perturbed image, and the constraint $||\Delta x||_p \leq \varepsilon$ ensures that perturbation is at most ε, which is a small and exogenously specified bound (in our experiments, at most 0.002 for any norm). Additionally, since image pixels are normalized in the $[0,1]$ interval, we also ensure that introduced perturbations result in valid images by adding the constraint that $x_0 + \Delta x \in [0,1]^n$. If our goal is to minimize, rather than maximize predicted age, the objective becomes minimization rather than maximization.

Since the optimization problem (5.1) is challenging as stated, we use heuristic approaches based on those introduced in prior literature for solving this problem [115]. As the specific approaches are tailored to the norm which measures the magnitude of the introduced perturbation, we next present such approaches for each norm.

5.2.1.1 The l_∞ Attack

Our approach to implementing the l_∞ norm attack is based on FGSM [39] and its subsequent iterative variation [61]. The idea behind the approach is to approximate the objective function $F(x_0 + \Delta x) \approx \nabla F(x_0)\Delta x + F(x_0)$. The optimal solution to this linearized objective is then $\Delta x = \varepsilon \text{ sign}(\nabla F(x_0))$. Extending this idea to an iterative variant, with N the number of iterations, we can take steps of size ε/N, where each step computes Δx using the gradient sign approach starting from the previous iterate. Finally, if the total modification to x_0 ever leaves the interval $[0,1]$, it is clipped to remain feasible. The full algorithm is given in Algorithm 6.

Algorithm 6 Single Target l_∞

```
1: **input:** predictor $F$, $l_\infty$ distance $\varepsilon$, iteration steps $N$, original
2: **output:** adversarial perturbation $\Delta x$
3: $t \leftarrow 0, i \leftarrow 0$
4: $\alpha \leftarrow \varepsilon/N$
5: **while** $i < N$ **do**
6:     $t \leftarrow t + \alpha \cdot \text{sign}(\nabla F(x_0 + t))$
7:     $t \leftarrow \text{clip}_{[0,1]}(x_0 + t)$
8:     $i \leftarrow i + 1$
9: **end while**
10: **return** $\Delta x \leftarrow t$
```

In the algorithm, the statement $t \leftarrow \text{clip}_{[0,1]}(x_0 + t)$ clips the argument to stay in the $[0,1]$ interval, modifying t accordingly.

These ideas extend in a straightforward way to minimizing $F(x_0 + \Delta x)$.

5.2.1.2 The l_2 Attack

Our approach for generating adversarial perturbations with respect to the l_2 norm follows Szegedy *et al.* [109] and Carlini and Wagner [20].

The main idea is to replace the hard constraint that $||\delta x||_2 \leq \varepsilon$ with an associated penalty in the objective. Specifically, we rewrite Problem (5.1) as follows:

$$\begin{aligned} &\text{minimize}_{\Delta x} \quad -c \cdot F(x_0 + \Delta x) + \|\Delta x\|_2 \\ &\text{subject to:} \quad x_0 + \Delta x \in [0,1]^n \end{aligned} \tag{5.2}$$

The constant c is used to balance maximizing $F(x+\Delta x)$ and minimizing $||\Delta x||_2$. By updating c, we can then find a Δx which satisfies $||\Delta x||_2 \leq \varepsilon$ and maximizes $F(x+\Delta x)$.

To deal with the box constraint $0 \leq x_0 + \Delta x \leq 1$, we follow Carlini and Wagner [20] and apply a change-of-variables, introducing a new variable ω such that:

$$\begin{aligned} x_0 &= \frac{1}{2}(\tanh(\omega_0) + 1) \\ \Delta x &= \frac{1}{2}(\tanh(\omega_0 + \Delta\omega) - \tanh(\omega_0)) \end{aligned}$$

Since $-1 \leq tanh(\omega) \leq 1$, the constraint $0 \leq x_0 + \Delta x \leq 1$ is always satisfied. With this transformation, we optimize over ω, rather than Δx. The transformed optimization problem becomes

$$\text{minimize}_{\Delta\omega} - c \cdot F\left(\frac{1}{2}(\tanh(\omega_0 + \Delta\omega) + 1)\right) + \|\tanh(\omega_0 + \Delta\omega) - \tanh(\omega_0)\|_2 .$$

The algorithm is shown as in Algorithm 7. In this algorithm, the optimizer uses N steps to find the optimal solution with the specific constant c. Every time we run the optimizer, it would try to make the result of $-c \cdot F(x_0 + \Delta x) + \|\Delta x\|_2$ smaller at a certain learning rate. After each time we run the optimizer, we would check whether the l_2 distance is smaller than the ε we have set and compare $F(x_0 + \Delta x)$ with the current maximum result.

As before, the approach is straightforward to modify if we wish to minimize predicted age.

Algorithm 7 Single Target l_2 Attack

```
1: input image $x_0$, predictor $F$, $L_2$ distance $\varepsilon$, number of iterations $N$, number of iterations of binary search $m$
2: output adversarial perturbation $\Delta x$
3: initialize $x' \leftarrow x, c \leftarrow c_0, i \leftarrow 0$, $\omega_0 \leftarrow \tanh^{-1}(2x-1)$
4: while $i < m$ do
5:     flag $\leftarrow$ False
6:     optimizer $\leftarrow$ optimizer.minimize$(-c \cdot F\left(\frac{1}{2}(\tanh(\omega_0+\Delta\omega)+1)\right) + \|\tanh(\omega_0+\Delta\omega)-\tanh(\omega_0)\|_2)$
7:     while $j < N$ do
8:         $\Delta\omega \leftarrow$ optimizer.run_one_step
9:         $\Delta x \leftarrow \frac{1}{2}(\tanh(\omega_0+\Delta\omega)-\tanh(\omega_0))$
10:        if $\|\Delta x\|_2 < \varepsilon$ then
11:            flag $\leftarrow$ True
12:            if $F(x+\Delta x) > F(x')$ then
13:                $x' \leftarrow x+\Delta x$
14:            end if
15:        end if
16:        $j \leftarrow j+1$
17:    end while
18:    if flag then
19:        increase $c$
20:    else
21:        decrease $c$
22:    end if
23:    $i \leftarrow i+1$
24: end while
25: return $\Delta x \leftarrow x' - x_0$
```

5.2.1.3 The l_0 attack

In the l_0 attack, the goal is to introduce an adversarial perturbation by modifying fewer than ε pixels in the image. Our method for doing this uses the intuition that the pixels with higher absolute gradient value play a more important role in the prediction output. Consequently, we find the pixels with the maximum absolute value of the gradient, and try to modify the value of these to maximize or minimize the model prediction. We iteratively do this until the maximum l_0 distance is achieved (i.e., we reach the threshold number of pixels we can modify). This approach for maximizing the prediction is formalized in Algorithm 8. To minimize predicted age, the only difference is to modify the value of one pixel in each iteration to make the prediction smaller, rather than larger.

Algorithm 8 Single Target l_0 Attack

```
1: input: image $x_0$, predictor $F$, $l_0$ distance upper bound $\varepsilon$, possible values @for@ each
   pixel $V = [v_1, v_2, ...v_n]$
2: output: adversarial perturbation $\Delta x$
3: initialize $x' \leftarrow x_0, i \leftarrow 0$, $G \leftarrow \nabla_x F(x')$
4: while $i < \varepsilon$ do
5:     $pos \leftarrow argmax_k(|G_k|)$
6:     $G_{pos} \leftarrow 0$
7:     $flag \leftarrow False$
8:     for $k$ in $V$ do
9:         $x'' \leftarrow x'$
10:        $x''_{pos} \leftarrow k$
11:        if $f(x'') > f(x')$ then
12:            $x' \leftarrow x''$
13:            $flag \leftarrow True$
14:        end if
15:    end for
16:    if flag then
17:        $i \leftarrow i + 1/size(x)$
18:    end if
19: end while
20: return $\Delta x \leftarrow x' - x_0$
```

5.2.2 Generating Adversarial Perturbations for a Batch of Images

Our final discussion concerns a method for generating a single adversarial perturbation Δx for a batch of input images $\{x_0, x_1, ..., x_m\}$. We formalize it as solving the following optimization problem:

$$\begin{aligned} \text{Maximize} \quad & G(\Delta x) = \sum_{i=0}^{m} F(x_i + \Delta x) \\ \text{subject to:} \quad & ||\Delta x||_\infty \leq \varepsilon \end{aligned} \tag{5.3}$$

(Note that we restrict attention to l_∞-norm attacks in this case, to simplify discussion.)

We optimize the objective by extending the iterative gradient-sign method discussed in Section 5.2.1.3. The full algorithm is given in Algorithm 9.

Algorithm 9 l_∞ Attack for a batch of images

```
1: input: a batch of original images $\{x_0, x_1, ..., x_m\}$, predictor $F$, $l_\infty$ upper bound $\varepsilon$, num-
   ber of iterations $N$ $\varepsilon$, possible values @for@ each pixel $V = [v_1, v_2, ...v_n]$
2: output: adversarial perturbation $\Delta x$
3: $t \leftarrow 0, j \leftarrow 0$
4: $\alpha \leftarrow \varepsilon / N$
5: while $j < N$ do
6:     $t \leftarrow t + \alpha \cdot \text{sign}\left(\sum_{i=0}^{m} \nabla F(x_i + t)\right)$
7:     $j \leftarrow j + 1$
8: end while
9: $\Delta x \leftarrow t$
```

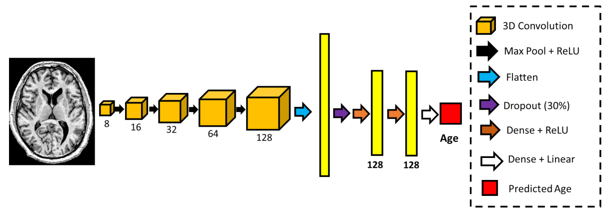
5.3 Results

Here we focus on data that are generally accessible and with an algorithm not likely to drive patient care to evaluate the effectiveness of such attacks in medical image processing settings in a way that does not violate clinical research ethics (as could be an issue, for example, if the target was a medical diagnosis). We expect that our results are generalizable, so long as similar image processing techniques are used.

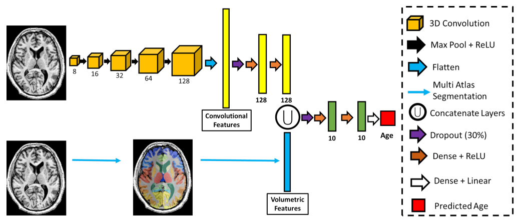
Our imaging dataset is an aggregate of 7 datasets with a total 3921 T1w 3D images from normal, healthy subjects. The data include subjects with ages ranging between 4 and 94 years old, with a mean age and standard deviation of 25.5 ± 18.6 years. Of the 3921 subjects, 54.2% were male and 45.8% were female. Data were also acquired from different sites so there is a difference in field strength, of which 71.5% of scans were acquired at 3 Tesla and 28.5% were acquired at 1.5 Tesla. ROI volumes, gender, and field strength were all used as input features for age prediction.

We consider two models for predicting age: 1) a conventional deep neural network, and 2) a hybrid (or context-aware) model which combines deep learning with image segmentation techniques. The conventional deep neural network model (*Conventional DNN*) takes a 3D brain MRI image as input and produces a subject's age as output. The architecture consists of five 3D convolution layers of increasing size followed by two densely connected layers and one output layer. The ReLU activation function was used for all hidden layers. The neural network was trained using a learning rate of 0.001. The structure of this model is shown in Figure 5.2a. The *context-aware model* has a similar structure to the conventional deep neural network model, with the exception that 132 volumetric features are introduced after the convolutional layers followed by two densely connected layers and, finally, the output layer. Volumetric estimates for 132 regions of interest in the brain (that is, each feature corresponds to the volume of a region of interest) were obtained using multi-atlas segmentation [56, 4]. The structure of the context-aware model is demonstrated in Figure 5.2b.

We consider three types of attacks which inject adversarial noise into an image: l_∞ attack, l_0 attack, and l_2 attack. All attacks limit the amount of noise being injected to ensure that is cannot be perceived by looking at the image, but differ in how they measure the amount of noise injected. The l_∞ attack considers modification to each pixel independently, and limits the amount any pixel can be modified. The l_0 attack limits the number of pixels modified. The l_2 attack limits the Euclidean norm of the injected adversarial perturbation.



(a) The structure of the conventional DNN



(b) The structure of the context-aware model

Figure 5.2: The models for brain age prediction

More precisely, define the perturbation as $\Delta x = (\Delta x_0, ..., \Delta x_N)$, where N is the number of pixels in the image. We define distortions in the respective norms as follows (we use slightly modified definitions here to make our results more intuitive):

$$l_\infty : \max_{i=0}^{N}\{\Delta x_i\}, \qquad l_2 : \sqrt{\frac{1}{N}\sum_{i=0}^{N}\Delta x_i^2}, \qquad l_0 : \frac{1}{N}\sum_{i=0}^{N}\mathbb{1}(\Delta x_i \neq 0). \tag{5.4}$$

The value of pixels in the original samples was normalized into range $[0,1]$.

The goal of adversarial perturbations is to either maximize or minimize the *predicted* (as opposed to *actual*) age. Since original predictions (without adversarial noise) are quite good (RMSE < 5.13 years), we use those as a baseline. We then measure the effectiveness of adversarial noise (in skewing the predictions) by *deviation*, defined as absolute change in predicted age:

$$deviation = |y' - y|, \tag{5.5}$$

where y is the original prediction (without noise), and y' the prediction after adversarial perturbation.

5.3.1 Conventional Deep Neural Networks are Fragile to Adversarial Perturbations of Medical Images

We first consider the impact of adversarial perturbations on a Conventional DNN, where we aim to maximize predicted age. Figure 5.3 illustrates this for the 19-year-old subject we discussed earlier, and presents results over the entire dataset, breaking these down by (originally predicted) age groups: $0-14, 15-25, 26-50, 51-65$, and > 65. As we can see from the illustration (images in the left column of the figure), we can cause the conventional DNN to predict age as 80 (rather than 19) using any of the three ways to quantify perturbation, with all three brain images looking indistinguishable from the original (in Figure 5.1a). As we would anticipate, l_0 perturbations are the most sparse, concentrated in parts of the image that have the greatest impact. A more systematic analysis in Figure 5.3 (plots in the right column) shows that age can be amplified nearly 70 years on average by adding perturbation with magnitude < 0.002 (for the normalized image) by any of the three measures. Interestingly, the most susceptible population is 15-25 year olds, across all three attack methods.

Similar trends are obtained if we inject adversarial noise in order to minimize predicted age (Figure 5.4). There appears to be little difference in which metric we use to bound adversarial perturbations: in all cases, with only a small amount of added noise, we can often reduce predicted age to nearly 0 for all age cohorts.

5.3.2 A Single Adversarial Perturbation Works for Large Batches of Images

While the most powerful attacks customize adversarial noise to each image, an alternative that may be more practical is to generate a single perturbation which can then be injected into any given image. We design such an attack, based on the l_∞-norm framework (which bounds the most any one pixel can be changed), and investigate its effectiveness as a function of the number of images that we target with a single attack. The attack maximizes

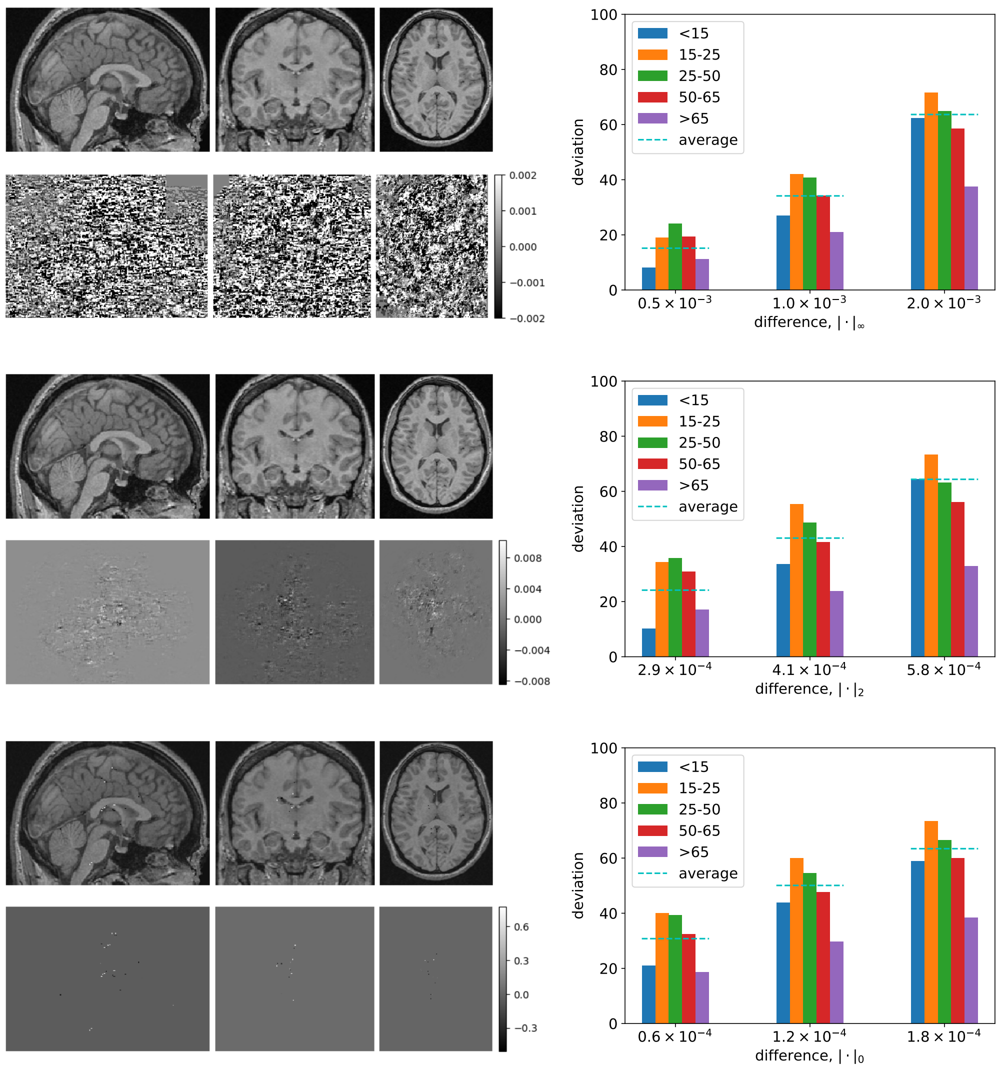


Figure 5.3: The adversarial perturbations that aim to maximize age. Images in the left column display the results of adversarial perturbations to the image of a 19-year-old subject in Figure 5.1a, using each of our three criteria for limiting the magnitude of the perturbation. The images in the top row of each of these correspond to the modified 2D slice images of the brain; immediately below is isolated noise that we add (amplified for visibility). In the right column we present general results of applying adversarial perturbations to images in our data (maximizing predicted age). In each plot, the x-axis is the limit of the amount of noise injected (where the noise bound is measured by each of our three l_p measures), while the y-axis is the corresponding impact, measured by deviation from original prediction. The first row of plots correspond to l_∞-bounded perturbations. The second row of plots represent results for l_2-bounded perturbations. The third row of plots are the results for l_0-bounded perturbations.

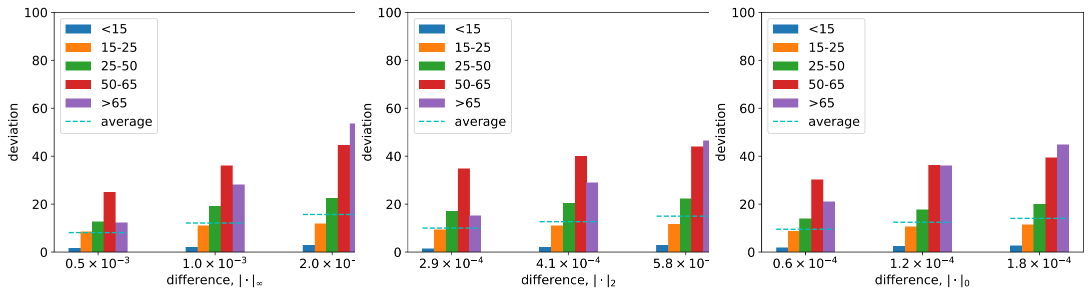


Figure 5.4: Attacks that aim to minimize predicted age. The x-axis limits the amount of noise injected, while the y-axis shows the corresponding impact, measured by deviation from original prediction. Left: adversarial perturbations bounded by the l_∞ metric. Middle: adversarial perturbations bounded by the l_2 metric. Right: adversarial perturbations bounded by the l_0 metric.

average predicted age for an entire batch of images.

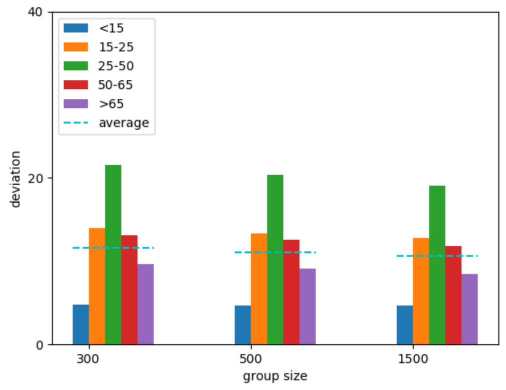


Figure 5.5: Attacking multiple images using the same adversarial perturbation for the conventional DNN model. The attack maximizes predicted age. We set the modification distance to 0.002. Group size corresponds to the number of images that we target with a single adversarial perturbation.

Figure 5.5 presents the results. Interestingly, once we consider more than 300 images in a batch, increasing the batch size has a relatively small impact on the effectiveness of adversarial perturbations. On average, perturbations result in an error in predicted of over 10 years (averaged over images in the batch, and random draws of batches), *even when we consider batches of 1500 images*. In the case of the most vulnerable cohort (<25-50 years old, in this case), the impact is over 20 years. Consequently, we can design highly effective adversarial perturbations that appear to be nearly universal.

5.3.3 Deep Learning with Volumetric Features based on Image Segmentation is Less Vulnerable

One of our most significant observations is not just that the conventional DNN model is vulnerable, but that incorporating features based on traditional multi-atlas image segmentation makes it *significantly less vulnerable* to adversarial perturbations.

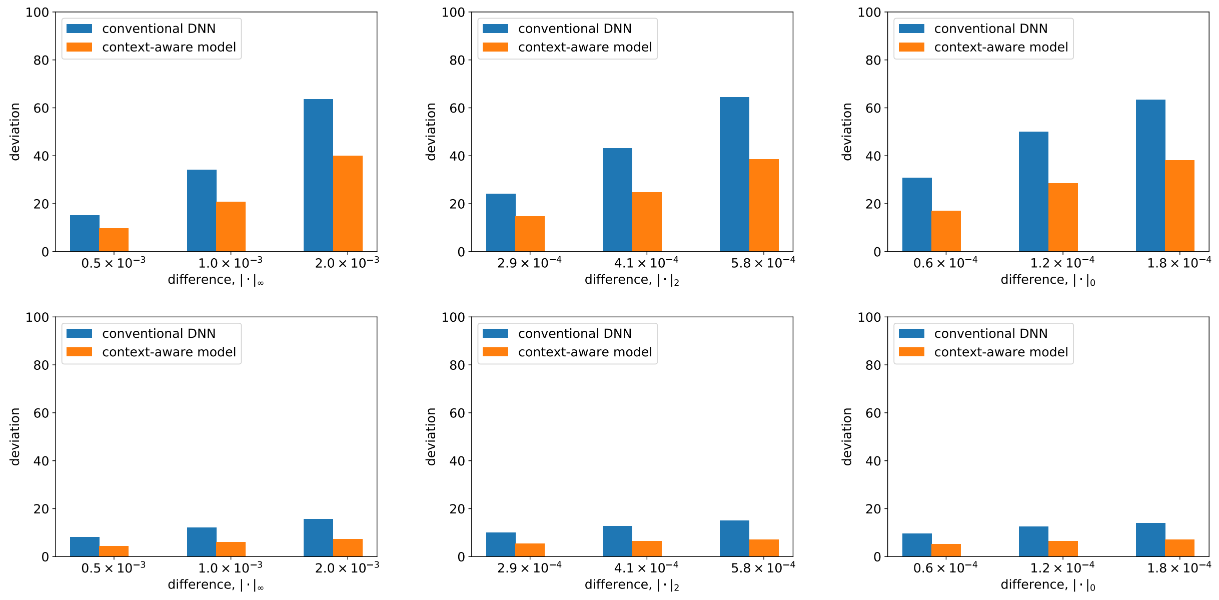


Figure 5.6: Adversarial perturbations designed for the context-aware model. The x-axis limits the amount of noise injected, while the y-axis shows the corresponding impact, measured by deviation from original prediction. Left plots correspond to l_∞ bounds (the most any one pixel can be changed) to measure impact. Middle plots correspond to l_2 bounds (Euclidean norm of the added noise). Right plots correspond to l_0 bounds (the fraction of pixels that can be changed). Top plots correspond to the objective of maximizing predicted age. Bottom plots are when we aim to minimize predicted age

Consider Figure 5.6 which presents the systematic analysis of the impact of adversarial perturbations on the context-aware model.[1] The difference with the conventional DNN is

[1]Because volumetric feature generation is extremely time consuming, these figures were generated by keeping such features invariant. In the Appendix, we present results with a small representative batch of images where we regenerated volumetric features after the adversarial perturbation, and our findings are

evident: in every case, the impact of the attack is significantly reduced, often by several factored. Nevertheless, it is not eliminated. For example, we can still introduce imperceptible noise (changing pixels by at most 0.2%) and in many cases increase predicted age by over 30 years.

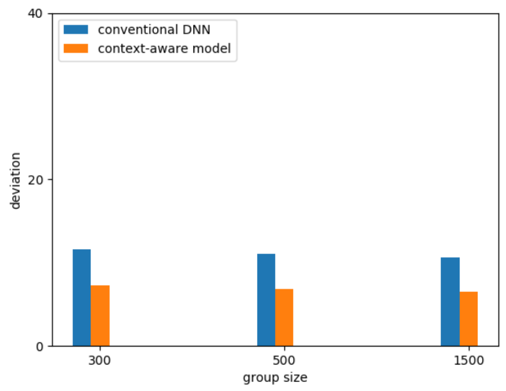


Figure 5.7: Attacking multiple images using the same adversarial perturbation for the context-aware model. The attack maximizes predicted age. We set the modification distance to 0.002. Group size corresponds to the number of images that we target with a single adversarial perturbation.

Similarly, we can observe that the impact of adversarial perturbations on image batches is significantly reduced for the context-aware model (Figure 5.7), where average impact on age drops from approximately 10 to just over 5 years. This drop is especially noteworthy since the adversarially induced error is now similar to the RMSE of the model prior to adversarial perturbations (which is just over 5 years).

5.4 Conclusion

Despite the increasing popularity of deep learning methods in medical imaging applications, our results suggest that significant concerns remain about robustness of these to adversarial perturbations to the environment. Such perturbations may arise simply due to unanticipated use cases or unusual patients, but may also be a product of actual tampering, for example, aiming to exploit introduced diagnostic bias for economic gain. While

largely consistent, since such features are relatively insensitive to small pixel-level perturbations.

DICOM supports robust security protocols, common software features are not uniformly implemented or applied [73], and data may be vulnerable to simple, direct manipulation on portable data systems (e.g., reliance on physical CD transport). While one may be skeptical about the practical relevance of adversarial perturbations to individual images, our results suggest that we can even generate a single perturbation which introduces significant bias into predictions made on *many* images. Moreover, the relatively opaque nature of deep learning models makes the problem particularly challenging, as erroneous predictions may be difficult to detect.

However, our results also suggest that a way to address fragility of deep learning models is by incorporating domain knowledge and more traditional multi-atlas image segmentation techniques. We believe that such methods introduce higher-level semantic information into the model which is significantly more robust to voxel-level image perturbations. While our experiments suggest that such a context-aware model may still be somewhat vulnerable to adversarial noise, it is significantly less so that a pure (conventional) deep neural network. Alternative approaches, such as adversarial retraining [108, 70], have also shown promise in significantly reducing vulnerability of machine learning algorithms, including deep learning, to adversarial perturbations. In the end, it is likely that a combination of techniques is needed to make deep learning sufficiently reliable for medical image processing applications.

Chapter 6

Feature Selection Games

6.1 Problem Overview

Despite recent advancements in machine learning algorithms, it has been shown that machine learning systems are inherently vulnerable to carefully-crafted attacks. In such cases, attackers can exploit the model's weaknesses at the test time by crafting malicious adversarial examples that causes intentional errors [12, 22, 45]. In this process, certain constraints need to be held to make sure the resulting adversarial examples are 'close' enough to the original inputs that make attacks to be undetectable. For example, one kind of attacks is called l_0 attacks, if the distance between the original inputs and adversarial examples are measured by the l_0 distance being the number of modified features. l_0 attacks first are posted by the image recognition community, leading a large family of variants, like Jacobian-based saliency map attack (JSMA) [107, 38] and one pixel attacks[106], which attract more interests from the other fields[96, 26].

On the other hand, defensive techniques attempting to mitigate the effects of adversarial example have been extensively investigated in the research literature, including adversarial examples predictions [42, 49, 77] and detection [76, 41, 44]. However, Carlini & Wagner [21] showed that most of these defenses could be defeated if the attacker crafts adversarial examples targeting the specific defense. To address such gap, varied game-theoretical approaches modeling the interactions between the learner and the adversary emerge [17, 16, 18], where the solution concept, equilibria, naturally yield the learner's best defense when facing the optimal attacks.

In this chapter, we focus on the issue of designing robust prediction algorithms against l_0 attacks. We propose a methodology to generate an optimal randomization scheme, a probability distribution over a collection of predictors with different feature subsets, increasing the system's robustness against l_0 attacks via a novel framework, feature selection games.

6.2 Model

In the traditional feature selection scheme, given a learning task, the learner aims to find the optimal feature subset minimizing prediction error. We extend this scenario to an adversarial learning setting by adding an additional player, an attacker that tries to achieve its malicious goals, formulated as a feature selection game. At the high level, the competition takes place on the feature space, where the learner chooses its learning model via feature selections at training time, and the attacker manipulates the test data through its selected feature subset at testing time. Below, we first briefly introduce several notions of feature selection. Then, we illustrate the full game.

Consider a learning task, where $D = \{x_i, y_i\}_{i=1}^{m} \subset \mathscr{X} \times \mathscr{Y}$ is the training time data set of m training samples drawn from an unknown distribution, where $\mathscr{X} \subset \mathbb{R}^d$ and $\mathscr{Y}$ denote respectively the input and output space of the learning task. The feature space is formed by d features, indexed by set $[d] \triangleq \{1, 2, ..., d\}$. Any selected feature subsets belong to the power set of $[d]$, denoted by $2^{[d]} \triangleq \{V | V \subset [d]\}$.

6.2.1 Preliminary

6.2.1.1 Feature selection

Feature selection can be understood as finding the feature subset of a certain size with good 'quality' which can be measured by a set function (e.g., a feature subset that leads to the largest possible generalization or equivalently to minimal risk) represented by $\omega : 2^{[d]} \to \mathbb{R}^{+}$. Therefore, the optimal feature subset S^{*} can be determined via

$$S^* = \arg \sup_{S \in 2^{[d]}} \omega(S; \{x_i, y_i\}_{i=1}^m)$$

$$\text{s.t.}:$$

$$|S| \leq d_0$$

where d_0 is the maximum number of selectable features.

Wrapper-based feature selection algorithms apply a search through the space of feature subsets using the estimated accuracy of a learning model as a measure of goodness of a particular feature subset. In this case, ω can be formulated in the following way

$$S^* = \arg \sup_{S \in 2^{[d]}} \omega_w(S; \tilde{T}, f^*, \{x_i, y_i\}_{i=1}^m) \tag{6.1a}$$

$$= \arg \sup_{S \in 2^{[d]}} \sum_{i=1}^{m} -L(f^*(x_i * \sigma(S), y_i)) \tag{6.1b}$$

$$\text{s.t.}:$$

$$f^* = \tilde{T}(\{x_i * \sigma(S), y_i\}_{i=1}^m) \tag{6.1c}$$

$$|S| \leq d_0 \tag{6.1d}$$

where $*$ denotes the point-wise product, L is the loss function, $\mathscr{F} \subset \mathbb{R}^{\mathbb{R}^d}$ indicates the family of classifying or regression functions and $\sigma : 2^{[d]} \to \{0,1\}^d$ represents a projection mapping a feature subset to a binary vector such that $\sigma(S)_i = 1$ indicates the ith feature is present in the subset S and $\sigma(S)_i = 0$ otherwise. Given a family $\mathscr{F}$, a feature subset S and the data set $\{x_i, y_i\}_{i=1}^m$, the learning algorithm $\tilde{T}$ outputs an optimal classifying or regression function $f^* \in \mathscr{F}$ trained on the data set using the selected subset feature.

6.2.1.2 Adversarial examples

Given a supervised learning model f trained on the data set $\{x_i, y_i\}_{i=1}^m$, we consider that the attacker can add a perturbation Δx_i to the input sample x_i and the corresponding prediction is changed from $f(x_i)$ to $f(x_i + \Delta x_i)$ such that is not accurate compared to the ground truth y_i. Moreover, the attacker ensures that the resulting adversarial example $x_i + \Delta x_i$ is close enough to the original input x_i to avoid detections. Formally, the adversarial example crafting procedure is described by following optimization problem:

$$\arg\max_{\Delta x_i} loss(f(x_i + \Delta x_i), y_i) \tag{6.2a}$$

s.t. :

$$||\Delta x_i||_t \leq B \tag{6.2b}$$

$$x_{min} \leq x_i + \Delta x_i \leq x_{max} \tag{6.2c}$$

where Constraint 6.2b limits the distance between the original input and the resulting adversarial examples, and Constraint 6.2c makes sure the adversarial input fit the input range of the original problem. By choosing different t for the norm of Δx_i, the adversarial manipulation can be formed in different ways.

In this work, we first focus on the cases when $t = 0$ called l_0 attacks so that the attacker is limited in the number of changeable features. To explicitly formulate the competitions between the learner and the attacker on the feature space, we use an alternative way to represent the l_0 attacks. Intuitively, the process of l_0 attacks contain two stages:

- The slave problem: given a feature subset, finding the optimal modification pattern limited on the selected subset, maximizing the loss.
- The master problem: finding the optimal feature subset such that it maximizes the resulting value of the slave problem.

Then, given a feature subset $S \in 2^{[d]}$, the slave problem can be formulated as,

$$\max_{\{\Delta x_i\}_{i=1}^m} \sum_{i=1}^{m} loss(f(x_i + \Delta x_i, y_i) \tag{6.3a}$$

s.t. :

$$\forall i, (x_{min} - x_i) * \sigma(S) \leq \Delta x_i \leq (x_{max} - x_i) * \sigma(S) \tag{6.3b}$$

and the master problem becomes,

$$\arg\max_{S \in 2^{[d]}} \max_{\{\Delta x_i\}_{i=1}^m} \sum_{i=1}^{m} loss(f(x_i + \Delta x_i * \sigma(S), y_i) \tag{6.4a}$$

s.t. :

$$|S| \leq B \tag{6.4b}$$

In this chapter, we use $K(S)$ ($K : 2^{[d]} \to \mathbb{R}^d$) to represent the adversarial perturbation yielded by equation 6.3a given the feature subset S.

6.2.2 Feature Selection Game

We begin by describing the feature selection games (FSG) as a two-player zero-sum game with complete information. A FSG $(\mathscr{A}, \mathscr{L}, U)$ is defined as following,

- **Players:** The attacker with strategy space $\mathscr{A}$ and utility function U_a, and the learner with strategy space $\mathscr{L}$ and utility function U_l.
- **Strategies:** A pure strategy for each player is a feature subset (a subset of $2^{[d]}$) so that the attacker injects the adversarial perturbations through the targeted features and the learner optimize its model via feature selections. We further assume that the maximum number of selectable features for the learner is c_l, the maximum number of changeable features for the attacker is c_a, and $c_a < c_l \leq d$. Therefore, the pure strategy spaces for both players are two uniform matroids, where

$\mathscr{A} = \{S | S \in 2^{[d]}, |S| \leq c_a\}$ and $\mathscr{L} = \{S | S \in 2^{[d]}, |S| \leq c_l\}$.

- **Utility functions:** The utility function for both players is the robust loss function, where the attacker is the maximizer and the learner is the minimizer. Let set functions $U_1 : \mathscr{A} \times \mathscr{L} \rightarrow \mathbb{R}^+$ and $U_2 : \mathscr{L} \rightarrow \mathbb{R}^+$ be the learner's losses with and without the injection of adversarial perturbations. We define $U(A,L) = \mu U_1(A,L) + (1-\mu)U_2(L)$ as the robust loss function, where μ is a trade-off parameter.

- **Solution Concepts:** We consider the situations where both players move simultaneously and the solution concept, mixed strategy Nash Equilibrium(NE). A mixed strategies is a distribution indicating an assignments of a probability to each pure strategy, which allows both players to randomize their actions. Let $\Delta_{\mathscr{A}}$ and $\Delta_{\mathscr{L}}$ be the simplexes spanned by the attacker's and the learner's pure strategies so that $a \in \Delta_{\mathscr{A}}$ and $l \in \Delta_{\mathscr{L}}$ are the attacker's and the learner's mixed strategies, where a_A, l_L is the probability that the attacker chooses strategy A and the learner chooses strategy L, and $\sum_{A \in \mathscr{A}} a_A = \sum_{L \in \mathscr{L}} l_L = 1$. Therefore, given a pair of mixed strategy (a,l), the payoff are the expected utilities $\overline{U}(a,l) = \mathbf{E}_{(A,L)\sim(a,l)} U(A,L)$[1]. A pair of mixed strategy (a^*, l^*) is called mixed strategy NE if and only if (a^*, l^*) is yielded by the following minmax problem,

$$\min_{l \in \Delta_{\mathscr{L}}} \max_{a \in \Delta_{\mathscr{A}}} \overline{U}(a,l) \tag{6.5}$$

  In this cases, both players cannot benefit from unilaterally rotating from its current strategy, which yields the optimally robust solution for the learner.

6.2.3 Wrapper Approach Based FSGs with Pre-trained Learning Models

Given the general framework of FSGs defined in the last section, we now deal with more concrete models by considering the specific loss functions U_1 and U_2 that are yielded

[1]In this work, for any utility function $U : \mathscr{T} \rightarrow \mathbb{R}^+, \overline{U} : \Delta_{\mathscr{T}} \rightarrow \mathbb{R}^+$ is the expected value of U given a point in the simplex $\Delta_{\mathscr{T}}$.

from different feature selection and attack approaches.

We first illustrate such a FSG based on wrapper approaches with pre-trained learning models, where both the learner and the attacker consider specific learning models and attacks, and the payoffs are evaluated by the exact performance of a given learning task. Wrapper approach based FSGs are computationally intensive, but intuitive and usually provide precise solutions for a particular type of learning models. We first consider the cases that the training data is known by both players, and the learner has a branch of pre-trained learning models with fixed parameters corresponding to training data set and selected feature subsets., and such assumptions are relaxed in the later sections.

Suppose a specific family of learning model $\mathscr{F}$, the attacker's and the learner's pure strategy space $\mathscr{A}$ and $\mathscr{L}$, training data set $\{x_i, y_i\}_{i=1}^m$, and the learning algorithm $\tilde{T}$. Assume both players have complete information on T and $\{x_i, y_i\}_{i=1}^m$ and the training and testing data have the same distribution. The payoffs of the game $U(A,L), \forall A \in \mathscr{A}, \forall L \in \mathscr{L}$, are constructed as the follow. At the training phase, for each pure strategy $L \in \mathscr{L}$, the learner applies its selected feature subset L and chooses an optimal learning model f_L^* yielded by the learning algorithm $\tilde{T}$ which minimizes the loss over the augmented training set $\{x_i * \sigma(L), y_i\}_{i=1}^m$. After that, the learner's loss on the clean training set is

$$U_2(L) = \sum_{i=1}^{m} loss(f_L^*(x_i * \sigma(L)), y_i)$$

while the loss on the adversarial testing set can be represented by

$$U_1(A,L) = \sum_{i=1}^{m} loss(f_L^*(x_i + K(A)) * \sigma(L), y_i)$$

Once the benefit functions U_1 and U_2 are specified, we get a payoff matrix $\forall A \in \mathscr{A} \forall L \in \mathscr{L}, U(A,L)$ and the problem of finding the mixed strategy NE defined by equation (6.5) can be solved via linear programming. However, such LP is difficult to be solved since there are exponential number of variables and constraints caused by the number of player's strategies.

To address the above problem, we consider the double oracle algorithm which starts at initial player's pure strategy spaces, iteratively solves a game within current strategy spaces, and enlarges it by adding the pure strategies via player's best-response oracles. The execution continues until convergence is detected, which means the learner's (attacker's) best-response oracle does not generate a pure strategy that is better than the learner's (attacker's) strategies in the support of the current equilibrium. Let $\mathscr{L}_0$ and $\mathscr{A}_0$ be the learner's and the attacker's initial pure strategy sets, $\mathscr{L}_t$ and $\mathscr{A}_t$ be the ones at the tth iteration, and (a_t^*, l_t^*) be the mixed strategy NE corresponding to the joint space of pure strategy $(\mathscr{A}_t, \mathscr{L}_t)$. At the $(t+1)$-th iteration, given the attacker's current strategy a_t^*, the learner's oracle yields its best-response,

$$L_{t+1}^* = \arg\min_{L \in \mathscr{L}} \overline{U}(a_t^*, L)$$

Similarly, the attacker's best-response can be calculated via

$$A_{t+1}^* = \arg\max_{A \in \mathscr{A}} \overline{U}(A, l_t^*)$$

When $L_{t+1}^* \in \mathscr{L}_t$ and $A_{t+1}^* \in \mathscr{A}_t$, the algorithm converges and returns the mixed strategy NE $(a^*, l^*) = (a_t^*, l_t^*)$. Otherwise, we form $\mathscr{L}_{t+1}$ and $\mathscr{A}_{t+1}$ by adding L_{t+1}^* to $\mathscr{L}_t$ and A_{t+1}^* to $\mathscr{A}_t$, and continues the procedure. Algorithm is formally described by Algorithm (10).

Thus, different learner's best-response oracles (which can be exact or approximate) yielding the exact (approximate) learner's best-response are the critical components shaping the game solving algorithms, which are described in the rest of this section.

Attacker's Oracle (AO) : The attacker's oracle AO is a function yielding the attacker's best response corresponding to a learner's mixed strategy. At the kth iteration, given a learner's support set $\mathscr{L}^{(k)}$ and the mixed strategy $l \in \Delta_{\mathscr{L}^{(k)}}$, for an input x, the expected

Algorithm 10 Double Oracle Algorithm

```
1: Initialize $\mathscr{A}^{(0)}$ by generating arbitrary candidate attacker's actions.
2: Initialize $\mathscr{L}^{(0)}$ by generating arbitrary candidate learner's actions.
3: $i \leftarrow 0$
4: **repeat**
5:     $(a,l) \leftarrow CoreLP(\mathscr{A}^{(i)}, \mathscr{L}^{(i)})$
6:     $\tilde{A} \leftarrow AO(l)$
7:     $\mathscr{A}^{(i+1)} \leftarrow \mathscr{A}^{(i)} \cup \{\tilde{A}\}$
8:     $\tilde{L} \leftarrow LO(a)$
9:     $\mathscr{L}^{(i+1)} \leftarrow \mathscr{L}^{(i)} \cup \{\tilde{L}\}$
10:    $i \leftarrow i+1$
11: **until** convergence
12: **return** $(a,l)$
```

learner's prediction can be represented by

$$f(x) = \sum_{L \in \mathscr{L}^{(k)}} l_L \cdot f_L(x * \sigma(L)) \tag{6.6}$$

To find an approximate attacker's best response, the attacker's oracle aims to find the adversarial permutation maximizing the adversarial training loss U_1 under l_0 constraints. In this work, the attacker's oracle applies the Jacobian-based Saliency Map Attack [86] on the ensemble predictor f.

Learner's Oracle (*LO*) **:** Similarly, the learner's oracle generates the learner's best response corresponding to an attacker's mixed strategy. At the kth iteration, given an attacker's support set $\mathscr{A}^{(k)}$ and the mixed strategy $a \in \Delta_{\mathscr{A}^{(k)}}$, when the learner plays a particular pure strategy L, the expected training loss can be represented by

$$\sum_{A \in \mathscr{A}^{(k)}} a_A \cdot \sum_{i=0}^{m} U(A,L) \tag{6.7}$$

To find the approximate learner's best response L^*, we consider the following greedy algorithm being similar to the forward feature selection. Start with an empty set. In each iteration, we keep adding the feature, which best improves the above-expected training loss until an addition of a new feature does not improve the performance or the number of selected features reaches to learner's budget.

6.2.4 Continuous FSGs

In previous sections, we illustrate the framework of FSGs modeling the interactions between the attacker and the learner with discrete action spaces. By sequentially generating adversarial perturbations and predictors via learner's and attacker's approximate oracles, FSGs can be approximately solved with an iterative procedure. We now demonstrate that such a framework can be extended to cases where both player's action spaces are continuous so that more general attack and defense strategies can be applied.

Considering attacker's actions in continuous space means that attacker's oracle directly solves the adversarial perturbation generating problem defined by 6.2, and maximizes the weighted loss (6.6) of the ensemble predictor corresponding to learner's mixed strategies. In this case, FSGM [38] and PGD [72] algorithms are used to generate the approximate attacker's best response.

On the other hand, the learner is allowed to select the optimum predictor rather than choosing feature subsets directly. In this work, given attacker's mixed strategies, we consider the learners oracles that retrain the predictor at each iteration, which minimized the expected utility.

6.3 Experiments

In this section, we show our experimental evaluation on the robustness of randomized predictors generated by FSGs.

Datasets considered: Breast cancer Wisconsin [94] and MNIST [66]. Breast cancer Wisconsin is a binary classification problem over the dataset computed from a digitized image of a fine needle aspirate of a breast mass. The features of the dataset describe characteristics of the cell nuclei present in the image and the number of features is 31. MNIST

Defense	None	FSGs	Adversarial training
Clean	.9822	.8947	.9349
Adversarial(JSMA)	.1953	.4069	.8402

Table 6.1: The results for wrapper approach based FSGs (the attacker's budget $||\Delta x||_0 = 2$)

Defense	None	FSGs	Adversarial training
Clean	.9822	.8888	.8934
Adversarial(JSMA)	.0532	.3747	.7869

Table 6.2: The results for wrapper approach based FSGs (the attacker's budget $||\Delta x||_0 = 4$)

is a classification problem over the image dataset containing handwritten digits labelled from 0 to 9. Each data has 28×28 gray scale pixels representing the features. We focus on the subset as a binary classification problem consisting of handwritten 7s and 9s as they have visually similar components.

Architecture: 3-layers fully connected neural network for breast cancer Wisconsin and 5-layers convolution neural network for MNIST.

Metric: *Original accuracy* and *adversarial accuracy*. The original accuracy is the percentage of examples which have been correctly classified in the original testing set, while *adversarial accuracy* represents the accuracy of the predictor under the presence of the adversarial perturbation.

Baselines: We compare our work to adversarial training [111], which is a method has been proven its robustness against several adversarial attacks in practice.

6.3.1 Wrapper Approach Based FSGs

We first present the results of wrapper approach based FSGs (detailed in Section 6.2.3) on breast cancer Wisconsin dataset. In the experiments, the defender's budget $c_l = 25$, the maximum number of iteration in double oracle algorithm is 100 and the trade-off parameter μ equals to 0.5. Table 6.1 and Table 6.2 shows the results when the attacker's budget $c_a = 2$ and $c_a = 4$.

Dataset	Breast cancer		
Defense	None	FSGs	Adversarial training
Clean	.9822	.9467	.9349
Adversarial(JSMA), $\varepsilon = .1$	.2248	.7278	.2485
Adversarial(JSMA), $\varepsilon = .3$	.0710	.3372	.1597
Adversarial(FGSM), $\|\|\Delta x\|\|_\infty = .1$	.2662	.8757	.4674
Adversarial(FGSM), $\|\|\Delta x\|\|_\infty = .3$	.0236	.8698	.4497
Adversarial(PGD), $\|\|\Delta x\|\|_\infty = .1$	.2071	.8343	.4970
Adversarial(PGD),$\|\|\Delta x\|\|_\infty = .3$	.0177	.7396	.4852

Table 6.3: The results for continuous FSGs with breast cancer dataset

Dataset	MNIST		
Defense	None	FSGs	Adversarial training
Clean	.99	.9544	.9407
Adversarial(JSMA), $\varepsilon = .1$	.3481	.8263	.6407
Adversarial(JSMA), $\varepsilon = .3$	.1124	.4401	.3650
Adversarial(FGSM), $\|\|\Delta x\|\|_\infty = .1$	.1206	.8805	.6924
Adversarial(FGSM), $\|\|\Delta x\|\|_\infty = .3$	.061	.8681	.6020
Adversarial(PGD), $\|\|\Delta x\|\|_\infty = .1$	.0988	.8596	.5940
Adversarial(PGD),$\|\|\Delta x\|\|_\infty = .3$	.0522	.8117	.5122

Table 6.4: The results for continuous FSGs with MNIST dataset

Although the predictor without any defense strategy gets the highest accuracy on the clean data set in all cases, the predictors yielded by FSGs and adversarial training get a huge advantage on adversarial accuracy with tiny sacrificing on the original accuracy. However, the adversarial training still has better performance than FSGs (because adversarial training can freely choose predictor's parameters, while FSGs is limited its action space on choosing feature subset).

6.3.2 Continuous FSGs

We now demonstrate the results of continuous FSGs (detailed in section 6.2.4) on both breast cancer Wisconsin and MNIST dataset. In the experiments, the maximum number of iteration in the double oracle algorithm is 100, and the trade-off parameter μ equals 0.5. In the training phase, we choose the PGD algorithm with budget $||\Delta x||_\infty = .3$ as the adversarial example generator for both FSGs (such that the PGD algorithm generates the optimal attack for the attacker's oracle) and adversarial training. In the testing phase, we consider the adversarial accuracy of the resulting predictor when facing various adversarial example generating algorithms, including JSMA, FGSM, and PGD. Let $\varepsilon = \frac{||\Delta x||_0}{\text{total number of features}}$ be the regularized attacker's budget in JSMA. The result is presented by Table 6.3 and Table 6.4.

We find that both continuous FSGs and adversarial training are greatly effective against adversarial examples. However, in this case, continuous FSGs perform better than adversarial training, probably due to the same reason that the predictor yielded by FSGs has larger parameter space than the one yielded by adversarial training (because the one yielded by FSGs is assembled by a group of predictors). It may also explain why the performance gap between FSGs and adversarial training is smaller on MNIST (because the parameter space of the base model for MNIST is larger).

6.4 Conclusion

The chapter presents a game-theoretical framework to design robust predictors migrating the effects of adversarial examples with a limited number of changeable features. We construct Feature Selection Games, where the attacker's and the learner's pure strategy spaces are the possible feature subsets they can choose. In this case, the solution concept, mixed strategy Nash equilibrium, reveals the optimal way for the learner to deploy randomization on a set of predictors with different feature subsets. To address the scalability issue, we further present an (approximate) game solver combining column generation technique, heuristic search, and compact representation of player's pure strategy spaces that accelerate the procedure of finding mixed strategy Nash equilibria. We empirically show our approach enhances the robustness of predictors with varied learning models in real-world classification and regression tasks, and investigate the trade-off between accuracy and robustness.

Chapter 7

Adversarial Gaussian Process Regression in Sensor Networks

7.1 Problem Overview

Cyber-physical systems are fundamental to operations of many safety critical systems, from power plants to autonomous cars. Such systems feature a control loop that maps sensor measurements to control decisions. In many applications, these decisions involve maintaining system state features, such as temperature and pressure, in a safe range, with anomaly detection employed to ensure that anomalous or malicious sensor measurements do not subvert system operation. Although anomaly detection has been studied in the literature, many existing approaches focus on non-adversarial setting. Our first contribution is a novel stealthy attack on systems featuring Gaussian Process regression (GPR) for anomaly detectiona popular choice for this task. Next, we pose the problem of robust GPR for anomaly detection as a Stackelberg game, and present a novel algorithmic approach for solving it. Our experimental evaluation demonstrates both the vulnerability of baseline systems to attack, as well as the increased robustness offered by our approach.

While Ghafouri et al. serves as a conceptual precursor, the approach cannot be applied to GPR-based anomaly detection, where the predictions of GPR are random variables, in contrast to the conventional regression approaches, which make deterministic predictions. Moreover, since the attack model of GRP-based anomaly detection is not defined previously, the robustness of anomaly detection as measured by the false anomaly detection alarm has not been addressed.

We consider the problem of vulnerability of CPS with GPR-based anomaly detection to stealthy attacks, as well as the problem of making such systems robust. First, we define the attack model on GPR-based anomaly detection. Due to the non-linearity and non-convexity in the attack optimization problem, we present a novel approach to find the optimal stealthy attack. Then, we further consider the associated robustness problem via a game theoretical framework, in which the defender considers sensor selection, in addition to the choice of detection thresholds, as a lever for making anomaly detection more robust to attacks. Finally, we evaluate our work using the Tennessee-Eastman Process Control System as a case study. As our experiments demonstrate, allowing the defender to select sensors significantly enhances our ability to limit the impact of stealthy attacks.

The chapter is organized as follows. Section 7.2 presents an overview of GPR in the context of anomaly detection for CPS. Section 7.3 then presents a novel stealthy attack on GPR, followed by a novel approach for robust GPR in Section 7.4. Finally, Section 7.5 presents experimental results, showing both the considerable impact that our attack can have despite conservative anomaly detection thresholds, and the significantly increased robustness to attack achieved by our approach for robust GPR-based anomaly detection.[1]

7.2 Anomaly Detection with Gaussian Process Regression

Since sensors can be faulty, it is critical to ensure that measurement errors do not impact the control loop. An important class of approaches for accomplishing this is regression-based anomaly detection, where a detected anomaly is either rapidly corrected, or alternative inputs (such as predicted, rather than measured sensor values) are used in the control loop. We start by describing this class of approaches generically, given a collection of L sensors.

Let $\mathbf{y} = (y_1, ..., y_L)$ represent a vector of sensor measurements. For each sensor s, a predictor $f_s(\tilde{\mathbf{y}}_{-s})$ is learned from past data of joint measurements of all sensors (which is assumed to be normal), which maps from observed readings of sensors other than s, $\tilde{\mathbf{y}}_{-s}$, to a predicted reading of sensor s, $\hat{y}_s$. The detection system then compares the difference between the predicted and observed measurements, $|\hat{y}_s - \tilde{y}_s|$, and triggers an alarm when this difference is large.

[1] Source code: https://www.dropbox.com/sh/guhnr8a7awtghre/AADxGU4z0isogccPeTUE4Mgfa?dl=0

Next, we describe how the Gaussian Process can be used in anomaly detection. The key advantage of the Gaussian Process over alternatives is that it directly captures variance of the prediction and, consequently, enables a principled approach to judge anomalies based on confidence intervals.

Consider a zero-mean Gaussian Process $y(x) \sim \mathscr{GP}(0, \mathscr{K}(x,x')) \in \mathbb{R}$, where $x, x' \in \mathbb{R}^d$ and $\mathscr{K}$ is a covariance function (For example, the squared exponential kernel, $exp(-\lambda||x - x'||_2^2)$). Suppose a training set D has n observations, and $D = \{(x_i, y_i)|i = 1, ..., n\}$. Let $\mathbf{x} = col(x_1, ..., x_n) \in \mathbb{R}^{n\times d}$ and $\mathbf{y} = [y_1, ..., y_n]^T \in \mathbb{R}^n$. Suppose the observations $\mathbf{y}$ is Gaussian

$$\mathbf{y} \sim \mathscr{N}(0, K(\mathbf{x}))$$

where $K(\mathbf{x}) \in \mathbb{R}^{n\times n}$ is a covariance matrix, $K_{i,j}(\mathbf{x}) = \mathscr{K}(x_i, x_j)$. Given a new point x_*, we aim to predict the value $y_*|D$. According to the fact that $y(x)$ is a Gaussian process,

$$\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} \sim \mathscr{N}\left(0, \left[\begin{array}{c|c} K(\mathbf{x}) & \mathbf{k}_*(\mathbf{x}, x_*) \\ \hline \mathbf{k}_*(\mathbf{x}, x_*)^T & k_{**}(x_*) \end{array}\right]\right) \tag{7.1}$$

where, $\mathbf{k}_*(\mathbf{x}, x_*) = [\mathscr{K}(x_1, x_*), ..., \mathscr{K}(x_n, x_*)]$, and $k_{**}(x_*) = \mathscr{K}(x_*, x_*)$. Then, the prediction can be made by, $y_*|D \sim N(\mu, \sigma^2)$ where

$$\mu = \mathbf{k}_*^T(\mathbf{x}, x_*)K(\mathbf{x})^{-1} \cdot \mathbf{y} \tag{7.2}$$

$$\sigma = -\mathbf{k}_*^T(\mathbf{x}, x_*)K^{-1}(\mathbf{x})\mathbf{k}_*^T(\mathbf{x}, x_*) + k_{**}(x_*) \tag{7.3}$$

Therefore, given a collection of historical observations of the system, the predictor f_s for each sensor s can be model via above process. In the running time, for each sensor s, given the observed reading, $(\tilde{y}_s, \tilde{\mathbf{y}}_{-s})$. The predictions $\hat{y}_s$ can be calculated from $f_s(\tilde{\mathbf{y}}_{-s})$, which are Gaussian and determined by (7.2) and (7.3). Then, we can use the predictions and further check whether the reading $\tilde{y}_s$ is anomalous based on the confidence interval of $\hat{y}_s$. When $\tilde{y}_s$ is not inside the τ_s confidence interval of $\hat{y}_s$, $\tilde{y}_s$ is flagged as anomalous. Formally,

$\tilde{y}_s$ is suspicious, if $F(\tilde{y}_s) > \tau_s$, where $F(\cdot)$ is the corresponding conditional Gaussian cdf, or equivalently,

$$|\tilde{y}_s - \mu(\hat{y}_s)| > \sqrt{2}\sigma(\hat{y}_s) erf^{-1}(\tau_s) \tag{7.4}$$

7.3 Stealthy Attacks on Gaussian Process Anomaly Detection

Clearly, if the anomaly detector is deployed, the attacker can no longer attack with impunity. We now ask whether one can design attacks more carefully, so as to successfully and significantly change the sensor readings of target sensors while remaining stealthy—that is, avoiding detection by the GP-based anomaly detection system.

We formalize the problem of stealthy attacks on GP-based anomaly detection as an optimization problem. Our threat model is an attacker who can compromise up to H sensors, capturing the fact that integrity attacks (such as man-in-the-middle attack) on individual sensors can be technically quite challenging. However, once a sensor is successfully compromised, we allow the attacker to make arbitrary modifications to sensor values, within a basic normal value range (for example, pressure readings cannot be negative) $[y_s^{min}, y_s^{max}]$ for each sensor s that is easy for defenders to check (we can think of this range as imposing an additional simplistic anomaly detector independently for each sensor). Let S_c be the set of critical sensors; the attacker's goal is to maximize or minimize observed measurements of these sensors. We focus on minimizing here; maximization can be handled analogously since the objective is linear. The primary motivation for this goal is that sensor measurements, particularly when it comes to critical sensors, impact the control loop. This impact is often of the following form: the controller aims to keep the system state—say, pressure—in a safe range. If the sensor reading the pressure shows that pressure is too low, the controller will cause the pressure to rise to reach the safe range. Consequently, if the attacker can compromise the pressure sensor to read that the pressure is low, it effectively causes the controller to increase pressure, potentially above the safe range!

The attacker's decision variables correspond to the change in sensor value for each sensor s, which we denote by $\Delta\tilde{y}_s$. We omit the subscript s when we refer to the *vector* of changes to all sensors, and use $\Delta\tilde{\mathbf{y}}_{-s}$ to denote the vector of changes to measured values for all sensors other than s. We also denote by $\tilde{y}_s$ the original measured value of the sensor s, and, as above, $\tilde{\mathbf{y}}_{-s}$ is the vector of measurements of all sensors other than s, while $\tilde{\mathbf{y}}$ is the vector of all sensor measurements. Then the attacker's optimization problem can be represented as the following mathematical program:

$$\min_{s^* \in S_c} \min_{\Delta\tilde{\mathbf{y}}} \Delta\tilde{y}_{s^*} \tag{7.5a}$$

$$\text{s.t.}: \tag{7.5b}$$

$$\forall s, \quad |\tilde{y}_s + \Delta\tilde{y}_s - \mu(\hat{y}_s)| \leq \sqrt{2}\sigma(\hat{y}_s) erf^{-1}(\tau_s) \tag{7.5c}$$

$$\forall s, \quad \hat{y}_s = f_s(\tilde{\mathbf{y}}_{-s} + \Delta\tilde{\mathbf{y}}_{-s}) \tag{7.5d}$$

$$\forall s, \quad y_s^{min} \leq \Delta\tilde{y}_s + \tilde{y}_s \leq y_s^{max} \tag{7.5e}$$

$$||\Delta\tilde{\mathbf{y}}||_0 \leq H \tag{7.5f}$$

Here, Constraint 7.5c ensures that the attack is not flagged as anomalous; Constraint 7.5e requires that measured sensor values are feasible (e.g., physically realizable), and Constraint 7.5f ensures that at most H sensors are actually attacked. Observe that in this formulation, the first minimum over the critical sensors $s^* \in S_c$ can be eliminated as we can simply solve the optimization problem for each critical sensor independently, and then choose the solution with the largest impact (i.e., the smallest optimal $\Delta\tilde{y}_{s^*}$). We therefore henceforth focus on the simpler objective $\min_{\Delta\tilde{\mathbf{y}}} \Delta\tilde{y}_{s^*}$ for some critical sensor s^*.

Even with the above simplification, the mathematical program (7.5) is clearly quite challenging. The major complication which qualitatively distinguishes this problem from the prior work on attacking regression-based anomaly detection [37] is that the predictor $f_s(\cdot)$ is now a *random variable* (Gaussian Process). In Constraint 7.5d, this random variable is explicitly identified by $\hat{y}_s$, and the stealth constraint (Constraint 7.5c) uses its mean, $\mu(\hat{y}_s)$, as well as its variance $\sigma(\hat{y}_s)$. The most important consequence is that both the mean and the variance functions are potentially non-convex, as they depend on the potentially non-convex kernel function. This is further complicated by the non-convex l_0 constraint which ensures that at most H sensors are attacked.

In order to solve the non-convex optimization problem (7.5) we apply the feasible direction local search method, which is a variant of *feasible direction methods* developed by Zoutendijk [124]. At the high level, in each iteration of this iterative method, we start with a feasible solution computed in the previous iteration and find a descent direction. We then move in the feasible space slightly in the descent direction to obtain a new solution. In this move, we first ignore the l_0 constraint 7.5f. Then, we find the closest feasible solution for which the l_0 constraint also holds.

Let D be the feasible domain formed by (7.5c) - (7.5e), and D' be the one formed by (7.5c) - (7.5f). We rewrite the feasible domain D as an abstract set of inequalities

$$D = \{g_j(\Delta\tilde{\mathbf{y}}) \geq 0, j = 1, ..., J\}$$

where $g_j(\cdot)$ are differentiable functions.

Step 1: For each iteration, $\Delta\tilde{\mathbf{y}}^{(k)}$ is a feasible solution in the kth step. We use following LP to find a locally feasible descent direction of the optimization problem (7.5) in the kth step:

$$\max_{\mathbf{d}^{(k)}} \alpha \tag{7.6a}$$

$$\text{s.t. :} \tag{7.6b}$$

$$\mathbf{e}_{s^*} \cdot \mathbf{d}^{(k)} \leq -\alpha \tag{7.6c}$$

$$\forall j = 1, ..., J, g_j(\Delta\tilde{\mathbf{y}}^{(k)}) + \nabla g_i(\Delta\tilde{\mathbf{y}}^{(k)}) \cdot \mathbf{d}^{(k)} \geq \alpha, \tag{7.6d}$$

where $\mathbf{e}_s = (\underbrace{0, ..., 1}_{s}, ..., 0)$ is the sth unit vector of the standard Euclidean basis and $\mathbf{d}^{(k)}$ is the direction we aim to find. When $\alpha \geq 0$, Objective 7.6a and Constraint 7.6c guarantee that the optimal solution of this LP yields a descent direction. When $\Delta\tilde{\mathbf{y}}^{(k)}$ reaches one of the edges of the feasible domain D, there is a small positive ε such that $\exists j, 0 < g_j(\Delta\tilde{\mathbf{y}}^{(k)}) < \varepsilon$. In this case, the term $\nabla g_j(\Delta\tilde{\mathbf{y}}^{(k)}) \cdot \mathbf{d}^{(k)}$ in Constraint 7.6d forces the computed direction to follow the angle into the interior of the feasible domain. If $\alpha \leq 0$, the iteration terminates, since such a direction does not exist.

Step 2: Find the solutions along the feasible descent direction in the feasible domain D. Let $\Delta\tilde{\mathbf{y}}^{(k)} + \beta_{max}\mathbf{d}^{(k)}$ be the point where some constraints are first activated such that β_{max} is the maximal length that the current iteration can climb along the direction $\mathbf{d}^{(k)}$. More precisely,

$$\beta_{max} = \min\{\beta | g_j(\Delta\tilde{\mathbf{y}}^{(k)} + \beta\mathbf{d}^{(k)}) = 0, j = 1, ..., J \text{ and } \beta \geq 0\}$$

Thus, the solution set is the segment from $\Delta\tilde{\mathbf{y}}^{(k)}$ to $\Delta\tilde{\mathbf{y}}^{(k)} + \beta_{max}\mathbf{d}^{(k)}$, represented by

$$\{\Delta\tilde{\mathbf{y}}^{(k)} + \beta\mathbf{d}^{(k)} | \beta \in [0, \beta_{max}]\}$$

We calculate the solution set as follows. Starting from $\Delta\tilde{\mathbf{y}}^{(k)}$, we find β_{max} by incrementally climbing along $\mathbf{d}^{(k)}$ with the step length ε. Assume β_{max} is found at p_{max} step ($\beta_{max} = p_{max}\varepsilon$), which forms the solution set

$$Z = \{\Delta\tilde{\mathbf{y}}^{(k)} + p\varepsilon\mathbf{d}^{(k)} | p = 0, 1, 2, ..., p_{max}\}.$$

Step 3: For the solutions found in step 2, we project them onto the domain satisfying the l_0 constraint and recheck their feasibility. More precisely, for each solution, we keep its H elements (at most H sensors are attacked) having the maximum sum of their absolute values and set the rest to zero. Consequently, the resulting solution is the closet point to the original one with respect to l_1 distance. Then, if the underlying solution is not in the feasible domain D', it will be discarded. Let $S = \{r_H(\mathbf{z}) = \arg\min_{\mathbf{z}'} ||\mathbf{z} - \mathbf{z}'||_1 : \mathbf{z} \in Z\}$ represent the set of projected solutions obtained in Step 3 corresponding to every solution

z in the solution set obtained in Step 2. Then, the optimal solution in the current iteration, $\Delta\tilde{\mathbf{y}}^{(k+1)}$ is the solution from this set S that most minimizes the objective value of the original problem.

The full algorithm for computing an approximately optimal attack is presented by Algorithm 11.

Algorithm 11 Optimal Stealthy Attack

```
Require: Step length $\varepsilon$, target sensor $s$.
 1: $k \leftarrow 0$
 2: $\Delta\tilde{\mathbf{y}}^{(k)} \leftarrow \mathbf{0}$
 3: while number of iteration $< n_{max}$ do
 4:     # step 1:
 5:     $\mathbf{d}^{(k)}, \alpha \leftarrow$ SLOVE_LP1$(\Delta\tilde{\mathbf{y}}^{(k)})$
 6:     if $\alpha < 0$ then
 7:         return $\Delta\tilde{\mathbf{y}}^{(k)}$
 8:     end if
 9:     # step 2:
10:     $p \leftarrow 0$
11:     loop
12:         $\mathbf{z}^{(p)} \leftarrow \Delta\tilde{\mathbf{y}}^{(k)} + p\varepsilon\mathbf{d}^{(k)}$
13:         if $\mathbf{z}^{(p)}$ does not fit (7.5c) - (7.5e) then
14:             $p_{max} \leftarrow p$
15:             Break.
16:         end if
17:         $p \leftarrow p + 1$
18:     end loop
19:     # step 3:
20:     for $p = 0, 1, ..., p_{max}$ do
21:         $\mathbf{z}'^{(p)} \leftarrow r_H(\mathbf{z}^{(p)})$
22:         if $\mathbf{z}'^{(p)}$ does not fit (7.5c) - (7.5f) then
23:             $\mathbf{z}'^{(p)} \leftarrow \Delta\tilde{\mathbf{y}}^{(k)}$
24:         end if
25:     end for
26:     $\Delta\tilde{\mathbf{y}}^{(k+1)} \leftarrow arg \min_{\mathbf{z}'^{(p)}} z_s'^{(p)}$
27:     $k \leftarrow k + 1$
28: end while
29: return $\Delta\tilde{\mathbf{y}}^{(k)}$
```

7.4 The Resilient Anomaly Detection System

Next, we consider the issue of hardening the anomaly detection system against attacks of the kind we discussed in the previous section. Suppose a CPS has a collection of critical sensors, the readings of which are considered as the direct inputs of system controller (e.g. the temperature reading for thermostats). To protect the system behavior from being adversely affected by the adversarially compromised readings of critical sensors while maintaining a low false alarm rate, the defender can leverage two types of decisions. First, the defender typically has many choices for sensor placement (indeed, the problem of optimal and resilient sensor placement has received independent attention in the literature [58, 59, 62]), and these choices can be made explicitly trading off resilience and false positive rate. Second, the defender can choose the confidence level for anomaly detectors that also trades off false alarm rate and resilience to attacks. Next we describe each of these decisions in more detail.

Sensor selection: The defender considers the following sensor selection problem. Let S be the set of N possible sensor locations, and suppose that the defender can place at most L sensors. Let $\theta \in \{0,1\}^N$ be a binary vector representing sensor placement decision with $\theta_i = 1$ if a sensor is placed at location i and $\theta_i = 0$ otherwise. We further assume that the critical sensors are always selected and $\theta_{i \in S_c} \equiv 1$ (otherwise, the controller cannot work due to the lack of the direct inputs). The budget constraint can then be represented as $\sum_{i=1}^{N} \theta_i \leq L$.

Confidence level: the second set of decisions the defender makes is in choosing the confidence levels τ for the anomaly detectors. This choice directly translates into the trade off we wish to capture: a narrower confidence interval will lead to more false alarms, but will also tighten the space within which the attacker can implement a stealthy attack, thereby limiting attack impact.

7.4.1 Resilient Anomaly Detection as a Stackelberg Game

We model the interaction between the defender, charged with designing a resilient sensor and anomaly detection system, and the attacker who can execute an integrity attack against this system, as a Stackelberg game. In this game, the defender moves first, choosing where sensors are placed among the finite set of locations S, as well as the confidence thresholds τ of the Gaussian Process anomaly detection system. The attacker then computes an attack in response to these decisions. We seek a *Strong Stackelberg Equilibrium (SSE)* of this game, where the attacker breaks ties in the defender's favor, and the defender chooses an optimal set of decisions to commit to, accounting for the attacker's optimal response (given this tie-breaking rule). Next, we describe this game formally.

We first define the false alarm rate FA, and assume that sensor measurements $\mathbf{y}$ before the attack follow the Gaussian Process distribution. The false alarm at sensor s is an event that a normal measured sensor value is misclassified as an anomaly, indicated by the following threshold function:

$$thr_s(\mathbf{y},\theta,\tau_s) = \begin{cases} 1 & |y_s - \mu(\hat{y}_s)| > \sqrt{2}\sigma(\hat{y}_s)erf^{-1}(\tau_s) \\ 0 & \text{otherwise} \end{cases}$$

where $\hat{y}_s = f_s(\mathbf{y}_{-s},\theta)$. When $thr_s(\mathbf{y},\theta,\tau_s) = 1$, the actual reading $\mathbf{y}$ is misclassified as an anomaly and sensor s fires a false alarm. We further define that the system fires a false alarm if *any* of its sensors have a false alarm. To capture this, let $thr(\mathbf{y},\theta,\tau) = \bigvee_s thr_s(\mathbf{y},\theta,\tau_s)$ where $s \in \{i|\theta_i = 1\}$. Therefore, we define the false alarm rate of the overall system as $FA(\theta,\tau) = \mathbb{E}_{\mathbf{y}} thr(\mathbf{y},\theta,\tau)$. In practice, we assume that we sample the actual readings of the system in some time interval between $t = 0$ and $t = T$ and denote by $\{\mathbf{y}^{(t)}\}_{t=0}^{T} = \{\mathbf{y}^{(0)},...,\mathbf{y}^{(T)}\}$. Then, the false alarm rate is the average value of threshold function on the samples, which is

$$FA(\theta,\tau) = \frac{1}{1+T}\sum_{t=0}^{T} thr(\mathbf{y}^{(t)},\theta,\tau). \tag{7.7}$$

Now we construct the full game. The game $G = (\mathscr{A}, \mathscr{V}, \mathscr{I})$ consists of:

- Players, $\mathscr{I} = \{a, d\}$, where a is the attacker and d the defender.
- Joint action space of the two players, $\mathscr{A} = \mathscr{A}_d \times \mathscr{A}_a$. The defender aims to solve above decision-making problems and we denote defender's actions as (θ, τ). The defender's budget is limited by the upper bound of number of placed sensors, L, and the false alarm rate, FA_{max}. Thus, $\mathscr{A}_d = \{(\theta, \tau) | \sum\limits_i \theta_i \leq L, FA(\theta, \tau) \leq FA_{max}\}$. On the other hand, we make the worst-case assumption that the attacker knows the actual sensor readings $\mathbf{y}$ at the time of the attack. Then, the attacker needs to decide the attack patterns $\Delta\mathbf{y}$ based on the value of $\mathbf{y}$.
- The utility functions $\mathscr{V}_a, \mathscr{V}_d$. The utility of the attacker is the expected attack impact on the sampled readings $\{\mathbf{y}^{(t)}\}_{t=0}^T$, which is $\mathscr{V}_a(\{\Delta\mathbf{y}^{(t)}\}_{t=0}^T, \theta, \tau) = \frac{1}{T+1} \sum\limits_{t=0}^{T} \min\limits_{s \in S_c} \Delta y_s^{(t)}$. We assume that our game is zero-sum.

Since our game is zero-sum, its Nash equilibrium and SSE are equivalent and can be found using the following maxmin program:

$$\max_{\theta,\tau} \min_{\{\Delta\mathbf{y}^{(t)}\}_{t=0}^T} \mathscr{V}_d(\{\Delta\mathbf{y}^{(t)}\}_{t=0}^T, (\theta, \tau)) \tag{7.8a}$$

s.t. :

$$\sum_{t=0}^{N} \theta_t \leq L, \quad FA(\theta, \tau) \leq FA_{max} \tag{7.8b}$$

$$\{\mathbf{y}^{(t)}\}_{t=0}^T \quad \text{solve (7.5).} \tag{7.8c}$$

7.4.2 Computing an Approximately Optimal Defense

In this section, we present our approach to approximately solve the game defined above.

Let $\Delta\mathbf{y} = \mathscr{S}\mathscr{A}(\mathbf{y}, \theta, \tau)$ be result of the stealthy attack given sensor selection decision θ and confidence levels τ, where $\mathbf{y}$ is the underlying sensor value vector corresponding to θ.

The utility of the defender *given the attack* is then $\mathscr{V}_d = \frac{1}{T+1} \sum_{t=0,\ldots T} \min_{s \in S_c} \mathscr{S}\mathscr{A}(\mathbf{y}^{(t)}, \theta, \tau)_s$. We call this defender's problem as the *master problem*, and term the attacker's problem of computing a stealthy attack in response to the defender's decision, $\mathscr{S}\mathscr{A}(\mathbf{y}, \theta, \tau)$, the *slave problem*.

Clearly, exhaustively exploring the defender's options is intractable. Our approach for solving the defender's problem proceeds in two steps: first, an algorithm for finding confidence level thresholds *given* sensor placement decisions θ, and second, an algorithm for sensor placement which computes confidence levels as a subroutine. We begin with the former.

Our approach for finding confidence level thresholds is to start at an initial configuration $\tau^{(0)}$ with some small values $\tau_s^{(0)}, \forall s \in \{i | \theta^{(0)} = 1\}$, which yields high resilience to attacks, but at the cost of a high false alarm rate. Then, we iteratively find the sensor with the highest false alarm rate and increase the associated thresholds τ_s until the false alarm rate drops below the upper bound, taking advantage of the fact that both the defender's utility and the false alarm rate decrease monotonically. (see Algorithm 12).

Algorithm 12 optimal threshold, $OT(\theta)$

```
1: input sensor selection pattern $\theta$, initial threshold $\tau^{(0)}$
2: $FA^{(0)} \leftarrow FA(\theta, \tau^{(0)})$
3: $k \leftarrow 0$
4: # When the current $FA$ greater than the upper bound,
5: while $FA^{(k)} > FA_{max}$ do
6:     # Find the sensor with maximum number of false alarms.
7:     $t = arg \max_{i \in \{j | \theta_j = 1\}} \mathbb{E}_{y \sim p(Y)}(thr_i(y, \theta, \tau))$
8:     # Increase the threshold of confidence level.
9:     $\tau^{(k+1)} \leftarrow \tau^{(k)}$
10:    $\tau_t^{(k+1)} \leftarrow \tau_t^{(k)} + \tau_\varepsilon$
11:    $FA^{(k+1)} \leftarrow FA(\theta, \tau^{(k+1)})$
12:    $k \leftarrow k + 1$
13: end while
14: return $\tau^{(k)}$
```

Given the above algorithm for computing thresholds given sensor placement choices θ, we now tackle the problem of resilient sensor selection. For this, we use the Best-First search algorithm. Specifically, given a particular sensor selection decision θ, there are two possible actions:

- *Forward selection*: if the number selected sensors is below the budget, we can add a new unselected sensor, and
- *Backward elimination*: we can eliminate a selected sensor, if such sensor does not belong to the set of critical sensors.

In the algorithm, "best" is defined with respect to the objective value of a subset of selected sensors (which is never above the budget L), and the search proceeds for a specified number of iterations. The search algorithm (Algorithm 13)

7.5 Experiments

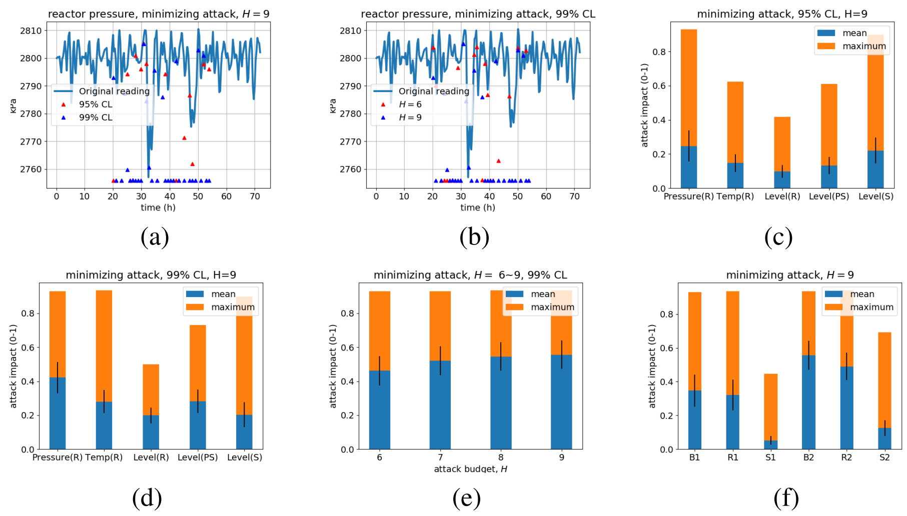


Figure 7.1: (a) and (b) show the effectiveness of our stealthy attacks on reactor pressure. (c) and (d) show the attack outcomes among critical sensors. (e) illustrates the attack outcomes with different attack budgets. (f) compares the performance between the resilient detectors and the baseline one. (Whiskers in the bar graphs indicate the 95% confidence level.)

We evaluate our approach using the Tennessee-Eastman process control system (TE-PCS).

Algorithm 13 optimal sensor selection pattern

```
1: input initial pattern $\theta^{(0)}$, visited states $V = \emptyset$, expanded states $E = \emptyset$
2: $\tau^{(0)} \leftarrow OT(\theta^{(0)})$
3: $\mathscr{V}^{(0)} \leftarrow \mathscr{V}_d(\{\mathbf{y}\}_{t=0}^{T}, \theta^{(0)}, \tau^{(0)})$
4: add $(\theta^{(0)}, \mathscr{V}^{(0)})$ to $E$ and $V$.
5: while number of iterations $\leq n_{max}$ or $E$ is empty do
6:     # Expand the state with the best performance
7:     $\theta, \mathscr{V} \leftarrow \arg\max_{(\theta', \mathscr{V}') \in E} \mathscr{V}'$
8:     remove $(\theta, \mathscr{V})$ from $E$
9:     # Forward selection
10:    if $\sum_i \theta_i < L$ then
11:        for $i \in \{i|\theta_i = 0\}$ do
12:            $\theta' \leftarrow \theta$
13:            $\theta'_i = 1$
14:            if $\theta' \in V$ then
15:                continue.
16:            end if
17:            $\tau' = OT(\theta')$
18:            add $(\theta', \mathscr{V}_d(\{\mathbf{y}\}_{t=0}^{T}, \theta', \tau'))$ to $E$ and $V$
19:        end for
20:    end if
21:    # Backward elimination
22:    if $\sum_i \theta_i - |S_c| > 0$ then
23:        for $i \in \{i|\theta_i = 1, i \notin S_c\}$ do
24:            $\theta' \leftarrow \theta$
25:            $\theta'_i = 0$
26:            if $\theta' \in V$ then
27:                continue.
28:            end if
29:            $\tau' = OT(\theta')$
30:            add $(\theta', \mathscr{V}_d(\{\mathbf{y}\}_{t=0}^{T}, \theta', \tau'))$ to $E$ and $V$
31:        end for
32:    end if
33: end while
34: # Return the optimal solution.
35: return $\arg\max_{(\theta, \mathscr{V}) \in V} \mathscr{V}$
```

Tennessee-Eastman Process Control System: TE-PCS is a widely studied industrial process, which consists of five main process components: a reactor, a separator, a stripper, a compressor and a mixer [32]. In this evaluation, we consider 5 critical sensors corresponding to safety constraints of TE-PCS (e.g., the upper bound of the pressure of the reaction container). We further assume that the system designer can select at most 15 sensors from 22 possibilities.

The model we use is the Simulink model of TE-PCS [6] with the implementation of the decentralized control law as proposed by [92]. For the anomaly detector, we first run simulation modeling the system operation for 72 hours and record the sensor measurements and control inputs. We take 225 timesteps periodically between 0-72 hours and record the all of sensor readings as the training set and train the collections of Gaussian process regression models under different sensor selection patterns. Then, from 20-60 hours, we record the sensor readings for every hour and get the testing set with 40 instances.

Stealthy Attacks: In the rest of the section, two baseline detectors are considered. Both baseline detectors have the same sensor selection pattern (xmeas(1) - xmeas(15) in Table 4, [32]) and fixed thresholds of confidence level (one is with 95% confidence level, called B1, and the other is with 99% confidence level, called B2). According to the definition of false alarm rate (7.7), B1 has 45% false alarm rate, while the false alarm rate of B2 is 20%. In this subsection, we perform our stealthy attacks on the baseline detectors and test the attack impacts.

First, we evaluate the effectiveness of our stealthy attacks on a single critical sensor (the reactor pressure). The results are presented in Figure 7.1a and Figure 7.1b. Figure 7.1a shows the attack impact on both B1 and B2 with a fixed attack budget ($H = 9$, that is, the attacker can attack at most 9 sensors), varying the confidence thresholds τ (95% vs. 99%). We can readily observe that the more conservative threshold (99%) in terms of limiting false positives is also significantly more susceptible to attacks compared to the more aggressive threshold. This exhibits the tradeoff we explicitly consider in designing robust anomaly detectors. Figure 7.1b shows the results on B2 with different attack budgets ($H = 6$ and $H = 9$, i.e., the attacker can attack at most 6 and 9 sensors, respectively). As one would expect, a stronger attacker (one with a larger budget) tends to effect a significantly greater impact on the system.

Next, we consider stealthy attacks on the aforementioned critical sensor set. Figure 7.1c and Figure 7.1d present the attack impact on both B1 and B2 among the 5 critical sensors. Figure 7.1e shows the attack impacts on the system for different attack budgets. We again observe similar trends: both the attacker budget and conservativeness of the anomaly detector confidence thresholds have a considerable influence on the impact of the attack. Nevertheless, impact is considerable throughout.

Resilient Detector: To evaluate our resilient detection approach, we fix the attack budget to $H = 9$, the maximum number of iterations of the Best-First search algorithm to 10, and consider 4 different resilient detectors,

- R1 and R2: R1 and R2 are the resilient detectors with optimized thresholds of confidence level and the original set of selected sensors (i.e., the same set of sensors as used by the baseline detectors).

- S1 and S2: S1 and S2 are the resilient detectors with optimized sensor selection patterns and optimized thresholds of confidence level.

To compare the resilient detectors with the baseline detectors B1 and B2, we let the upper bound of the false alarm rate of R1 and S1 equal to the false alarm rate of B1 (45%). Similarly, both R2 and S2 have the same upper bound of false alarm rate which equals to the false alarm rate of B2 (20%).

Figure 7.1f presents the results, which are quite instructive. While we see the general improvement in robustness (reduction in attack impact) moving fom Bx to Rx to Sx (with $x \in \{1,2\}$), it is clear that much of the final improvement is due to the careful choice about which sensors are used, with the tuning of thresholds given the original sensor placement providing only limited robustness. This observation serves as important motivation for jointly considering the problem of optimizing sensor and threshold selection in designing robust anomaly detectors, in contrast to prior work which considered each problem in isolation.

7.6 Conclusions

We considered a setting where a cyber-physical system (CPS) is monitored by a Gaussian process regression-based anomaly detection system. First, we presented a stealthy attack on this system that aims to maximize damage while appearing normal to the detector, presenting a novel approach to approximately solve this non-trivial optimization problem. Next, we presented a model of robust anomaly detection for CPS as a Stackelberg game, and developed a novel approach for solving this game, in which the defender can decide which sensors to use as well as how to set anomaly thresholds. Our experiments demonstrate that the attack can be quite effective on a baseline system, but our Stackelberg game solution approach is significantly more robust.

Chapter 8

Conclusion

8.1 Discussion

The central question of this thesis is ”how to forge resilient decision support systems in uncertain and adversarial environment?”. While resilience is considered as one of the general system’s attributes, the question is still difficult to be answered without taking into account its domain-specific definition.

In this thesis, we first consider the scenario of solving real-time control problems via simulation-based optimizations, where resilient decision support systems aim to provide rapid reactions to real-time events. We then introduce SBOaaS (simulation-based optimization as a service), a facility of cloud-based simulation services, which decompose the input problem into a group of parallel simulations and use the computing power through an anytime parallel optimization scheme. Our work admits significant flexibility in both time and computational resource constraints to obtain the best (but possibly sub-optimal) solutions given the available resources and time constraints on decisions, which can be seamlessly deployed to feed-back control loops.

Then, we investigate the scenario of multi-agent path planning (path planning games), where a collection of self-interested agents with motion uncertainty trend to find collision-free paths reaching their goals. In these cases, a resilient planner seeks the capability of balancing its performance and risk. Since such agents make their decisions in individual, potentially diverse ways, consequently, in order to study path planning games, we must take an economic, rather than a purely algorithmic, perspective on path planning. To this end, we first developed a novel mathematical programming method for computing a single-agent path plan, accounting for these two objectives, given fixed dynamic behavior (i.e., path plans) of all other agents, as well stochastic disturbances in the environment.

Next, we proposed a simple iterative algorithm for approximately computing Nash equilibria of path planning games, given the best response mathematical programs. Finally, we developed a novel mathematical program for computing a cooperative multi-agent path plan which optimally trades off efficiency and safety among all agents that is, again, taking the economic perspective on the multi-agent path planning problem. We numerically investigate path planning games through several case studies involving two and three agents. Our central observation is that as safety becomes more important to agents, a large gap opens up between safety achieved by a socially optimal and Nash equilibrium outcomes; in other words, Nash equilibria exhibit significantly more collisions than desirable by all agents. The main reason for this is that while each agent is concerned with safety, they only account in their objective for the impact of collisions on themselves, and not on other agents who crash along with them.

In the rest of this thesis, we pay our attention to machine learning enhanced decision support systems when facing malicious attacks. As a resilient learner aims to keep its performance stable after attacks, in this case, the specified property of resilience is robustness. We first study the problem of attracting medical image processing using adversarial noise. We described three approaches generating adversarial perturbations for a single image, and then presented their method that attacks a batch of images with a single perturbation. Our experiments show that a small amount of adversarial noise can lead to large predict errors in medical image processing, but more complicated model with context-information can reduce the effect of adversarial attacks. As it is shown that domain-specific knowledge is effective to improve the robustness, the next question is if it is possible for us to find the learner's optimal strategy in general cases. Then, we formulated feature selection game, a framework explicitly considering the learner's and the attacker's behaviors (where the learner applies the feature selection, and the attacker applies l_0 attacks), yields the optimal learner's strategy by maximizing the learner's utility in the worst case. By applying the double oracle algorithm, we addressed the computational issue of the game and extended both players' actions to more general cases by choosing appropriate oracles. Our experiments show that our approach is partially advance to state-of-the-art robust machine algorithms. As our previous concerns were focusing on the model containing a single artificial intelligence network, in the last part of this work, we considered a setting with the richer structure where a cyber-physical system is monitored by a Gaussian process regression-based anomaly detection system. First, we presented a stealthy attack on this system that aims to maximize damage while appearing normal to the detector, presenting a novel approach to approximately solve this non-trivial optimization problem. Next, we presented a model of robust anomaly detection for CPS as a Stackelberg game and developed a novel approach for solving this game, in which the defender can decide which sensors to use as well as how to set anomaly thresholds. Our experiments demonstrate that the attack can be quite effective on a baseline system, but our Stackelberg game solution approach is significantly more robust.

8.2 Future Work

8.2.1 Mechanism design in path planning game

The results in Chapter 4 show the gap in the performance between the multiple path planners working in centralized and decentralized manners, especially in the scenarios where agents are highly competitive. However, in large scale problems, such centralized planners are not feasible due to the exponential growth on joint action space with the number of agents. Hence, it is natural to raise the question, how can we manipulate the game to improve the overall performance with a collection of self-interested path planners.

On the other hand, the theory of algorithmic mechanism design [82] provides solid theoretical foundations to designing economic mechanisms or incentives, toward desired objectives, in strategic settings, where players act rationally. Therefore, the integration of mechanism design to path planning games is one of the interesting future directions.

8.2.2 Path planning game with multiple reinforcement learners

Another direction is the combination of reinforcement learning and path planning games. In this case, the game is formed by a collection of self-interested reinforcement learning agents learning their action preferences from the environment. While the traditional agents make their decisions after the planning process, the reinforcement learning agents apply their actions in a myopic way (because they make their preferred actions according to the previous observation), which leads to a considerable difference in the model's formulation.

BIBLIOGRAPHY

[1] Khaldoon Al-Zoubi and Gabriel Wainer. Distributed Simulation using RESTful Interoperability Simulation Environment (RISE) Middleware. In *Intelligence-Based Systems Engineering*, pages 129–157. Springer, 2011.

[2] Somaieh Amraee, Abbas Vafaei, Kamal Jamshidi, and Peyman Adibi. Anomaly detection and localization in crowded scenes using connected component analysis. *Multimedia Tools and Applications*, 77(12):14767–14782, 2018.

[3] Bo An, Milind Tambe, Fernando Ordonez, Eric Shieh, and Christopher Kiekintveld. Refinement of strong stackelberg equilibria in security games. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[4] Andrew J Asman and Bennett A Landman. Hierarchical performance estimation in the statistical label fusion framework. *Medical Image Analysis*, 18(7):1070–1081, 2014.

[5] Youakim Badr, Salim Hariri, AL-Nashif Youssif, and Erik Blasch. Resilient and trustworthy dynamic data-driven application systems (dddas) services for crisis management environments. *Procedia Computer Science*, 51:2623–2637, 2015.

[6] Andreas Bathelt, N Lawrence Ricker, and Mohieddine Jelali. Revision of the tennessee eastman process model. *IFAC-PapersOnLine*, 48(8):309–314, 2015.

[7] M Beckmann, CB McGuire, and CB Winsten. Studies in the Economics of Transportation. 1956.

[8] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo–simulation of urban mobility. In *The Third International Conference on Advances in System Simulation (SIMUL 2011), Barcelona, Spain*, 2011.

[9] Babak Ehteshami Bejnordi, Mitko Veta, Paul Johannes van Diest, Bram van Ginneken, Nico Karssemeijer, Geert Litjens, Jeroen AWM van der Laak, Meyke Hermsen, Quirine F Manson, Maschenka Balkenhol, et al. Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. *Journal of the American Medical Association*, 318(22):2199–2210, 2017.

[10] Camilo Bermudez, Andrew J Plassard, Shikha Chaganti, Yuankai Huo, Katherine E Aboud, Laurie E Cutting, Susan M Resnick, and Bennett A Landman. Anatomical context improves deep learning on the brain age estimation task. *Magnetic Resonance Imaging*, 2019.

[11] D.P. Bertsekas. *Nonlinear programming*. 1999. ISBN 1886529000. doi: 10.1137/1.9780898719383.

[12] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[13] Lars Blackmore, Hui Li, and Brian Williams. A probabilistic approach to optimal robust path planning with obstacles. In *American Control Conference*, 2006.

[14] D. Braess. Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung Operations Research - Recherche Opérationnelle*, 12(1):258–268, dec 1968. ISSN 0340-9422. doi: 10.1007/BF01918335.

[15] Michele Breton, Abderrahmane Alj, and Alain Haurie. Sequential stackelberg equilibria in two-person games. *Journal of Optimization Theory and Applications*, 59 (1):71–97, 1988.

[16] Michael Brückner and Tobias Scheffer. Stackelberg games for adversarial prediction problems. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 547–555. ACM, 2011.

[17] Michael Brückner, Christian Kanzow, and Tobias Scheffer. Static prediction games for adversarial learning problems. *Journal of Machine Learning Research*, 13(Sep): 2617–2654, 2012.

[18] Samuel Rota Bulò, Battista Biggio, Ignazio Pillai, Marcello Pelillo, and Fabio Roli. Randomized prediction games for adversarial machine learning. *IEEE transactions on neural networks and learning systems*, 28(11):2466–2478, 2016.

[19] Alvaro A Cárdenas, Saurabh Amin, Zong-Syun Lin, Yu-Lun Huang, Chi-Yen Huang, and Shankar Sastry. Attacks against process control systems: risk assessment, detection, and response. In *Proceedings of the 6th ACM symposium on information, computer and communications security*, pages 355–366. ACM, 2011.

[20] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, pages 39–57, 2017.

[21] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017.

[22] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[23] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 513–530, 2016.

[24] Mo Chen, Zhengyuan Zhou, and Claire J Tomlin. Multiplayer reach-avoid games via low dimensional solutions and maximum matching. In *American Control Conferenc*, pages 1444–1449, 2014.

[25] Mo Chen, Zhengyuan Zhou, and Claire J Tomlin. A path defense approach to the multiplayer reach-avoid game. In *Annual Conference on Decision and Control*, pages 2420–2426, 2014.

[26] Yize Chen, Yushi Tan, and Deepjyoti Deka. Is machine learning in power systems vulnerable? In *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–6. IEEE, 2018.

[27] Kai-Wen Cheng, Yie-Tarng Chen, and Wen-Hsien Fang. Gaussian process regression-based video anomaly detection and localization with hierarchical feature representation. *IEEE Transactions on Image Processing*, 24(12):5288–5301, 2015.

[28] Changbeom Choi, Kyung-Min Seo, and Tag Gon Kim. Dexsim: an experimental environment for distributed execution of replicated simulators using a concept of single-simulation multiple scenarios. *Simulation*, page 0037549713520251, 2014.

[29] WanKyoo Choi, HongSang Yoon, Kyungsu Kim, IlYong Chung, and SungJoo Lee. A traffic light controlling flc considering the traffic congestion. In *AFSS International Conference on Fuzzy Systems*, pages 69–75. Springer, 2002.

[30] James H Cole, Rudra PK Poudel, Dimosthenis Tsagkrasoulis, Matthan WA Caan, Claire Steves, Tim D Spector, and Giovanni Montana. Predicting brain age with deep learning from raw imaging data results in a reliable and heritable biomarker. *NeuroImage*, 163:115–124, 2017.

[31] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit

to. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 82–90. ACM, 2006.

[32] James J Downs and Ernest F Vogel. A plant-wide industrial process control problem. *Computers & chemical engineering*, 17(3):245–255, 1993.

[33] Yann Dujardin, Florence Boillot, Daniel Vanderpooten, and Pierre Vinant. Multi-objective and multimodal adaptive traffic light control on single junctions. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1361–1368. IEEE, 2011.

[34] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115, 2017.

[35] Peter Geibel and Fritz Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *J. Artif. Intell. Res.*, 24:81–108, 2005.

[36] Amin Ghafouri, Aron Laszka, Abhishek Dubey, and Xenofon Koutsoukos. Optimal detection of faulty traffic sensors used in route planning. In *Proceedings of the 2nd International Workshop on Science of Smart City Operations and Platforms Engineering*, pages 1–6. ACM, 2017.

[37] Amin Ghafouri, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. Adversarial regression for detecting attacks in cyber-physical systems. In *International Joint Conference on Artificial Intelligence*, pages 3769–3775, 2018.

[38] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[39] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harness-

ing adversarial examples. In *International Conference on Learning Representations*, 2015.

[40] Robert C Grande, Girish Chowdhary, and Jonathan P How. Experimental validation of bayesian nonparametric adaptive control using gaussian processes. *Journal of Aerospace Information Systems*, 11(9):565–578, 2014.

[41] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

[42] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

[43] Varun Gulshan, Lily Peng, Marc Coram, Martin C Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Journal of the American Medical Association*, 316(22):2402–2410, 2016.

[44] Dan Hendrycks and Kevin Gimpel. Early methods for detecting adversarial images. *arXiv preprint arXiv:1608.00530*, 2016.

[45] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and JD Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58. ACM, 2011.

[46] Yong K Hwang and Narendra Ahuja. Gross motion planninga survey. *ACM Computing Surveys (CSUR)*, 24(3):219–291, 1992.

[47] Manish Jain, Erim Kardes, Christopher Kiekintveld, Fernando Ordónez, and Milind

Tambe. Security games with arbitrary schedules: A branch and price approach. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[48] Manish Jain, Jason Tsai, James Pita, Christopher Kiekintveld, Shyamsunder Rathi, Milind Tambe, and Fernando Ordónez. Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces*, 40 (4):267–290, 2010.

[49] Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. Robust convolutional neural networks under adversarial noise. *arXiv preprint arXiv:1511.06306*, 2015.

[50] Anders Jonsson and Michael Rovatsos. Scaling up multiagent planning: A best-response approach. In *ICAPS*, 2011.

[51] Jaume Jordán, Alejandro Torreno, Mathijs de Weerdt, and Eva Onaindia. A better-response strategy for self-interested planning agents. *Applied Intelligence*, pages 1–21, 2017.

[52] Siddharth Joshi and Stephen Boyd. Sensor selection via convex optimization. *IEEE Transactions on Signal Processing*, 57(2):451–462, 2009.

[53] Khurum Nazir Junejo and Jonathan Goh. Behaviour-based attack detection and classification in cyber physical systems using machine learning. In *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security*, pages 34–43. ACM, 2016.

[54] Brian Kent, B Bruce Bare, Richard C Field, and Gordon A Bradley. Natural resource land management planning using large-scale linear programs: the usda forest service experience with forplan. *Operations Research*, 39(1):13–27, 1991.

[55] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. Computing optimal randomized resource allocations for massive

security games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 689–696. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[56] Arno Klein, Tito Dal Canton, Satrajit S Ghosh, Bennett Landman, Joel Lee, and Andrew Worth. Open labels: online feedback for a public resource of manually labeled brain images. In *Annual Meeting for the Organization of Human Brain Mapping*, 2010.

[57] Bin Kong, Xin Wang, Zhongyu Li, Qi Song, and Shaoting Zhang. Cancer metastasis detection via spatially structured deep network. In *International Conference on Information Processing in Medical Imaging*, pages 236–248. Springer, 2017.

[58] A. Krause, B. McMahan, C. Guestrin, and A. Gupta. Robust submodular observation selection. *Journal of Machine Learning Research*, 9:2761–2801, 2008.

[59] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian Processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

[60] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[61] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arxiv preprint*, abs/1607.02533, 2016.

[62] Aron Laszka, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. Resilient observation selection in adversarial settings. In *Conference on Decision and Control*, 2015.

[63] Aron Laszka, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. Resilient observation selection in adversarial settings. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 7416–7421. IEEE, 2015.

[64] Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[65] Steven M LaValle. Robot motion planning: A game-theoretic foundation. *Algorithmica*, 26(3-4):430–465, 2000.

[66] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

[67] Jay Lee, Behrad Bagheri, and Hung-An Kao. Recent advances and trends of cyber-physical systems and big data analytics in industrial informatics. In *International proceeding of int conference on industrial informatics (INDIN)*, pages 1–6, 2014.

[68] George Leitmann. On generalized stackelberg strategies. *Journal of optimization theory and applications*, 26(4):637–643, 1978.

[69] Roberto Leyva, Victor Sanchez, and Chang-Tsun Li. Video anomaly detection with compact feature sets for online performance. *IEEE Transactions on Image Processing*, 26(7):3463–3478, 2017.

[70] Bo Li and Yevgeniy Vorobeychik. Evasion-robust classification on binary domains. *ACM Transactions on Knowledge Discovery from Data*, 12(4):Article 50, 2018.

[71] Julia Lima Fleck and Christos G Cassandras. Infinitesimal perturbation analysis for quasi-dynamic traffic light controllers. In *Discrete Event Systems*, volume 12, pages 235–240, 2014.

[72] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[73] Fintan J McEvoy and Eiliv Svalastoga. Security of patient and study data associated

with dicom images when transferred using compact disc media. *Journal of Digital Imaging*, 22(1):65–70, 2009.

[74] Lucas Mearian. Self-driving cars could create 1gb of data a second. *Computerworld*, 23, 2013.

[75] Dirk Merkel. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux J.*, 2014(239), March 2014. ISSN 1075-3583. URL http://dl.acm.org/citation.cfm?id=2600239.2600241.

[76] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017.

[77] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 27–38. ACM, 2017.

[78] Patric Nader, Paul Honeine, and Pierre Beauseroy. lp-norms in one-class classification for intrusion detection in scada systems. *IEEE Trans. Industrial Informatics*, 10 (4):2308–2317, 2014.

[79] Kumpati S Narendra and Kannan Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on neural networks*, 1 (1):4–27, 1990.

[80] Nhan Nguyen and Mohammad Maifi Hasan Khan. A closed-loop context aware data acquisition and resource allocation framework for dynamic data driven applications systems (dddas) on the cloud. *Journal of Systems and Software*, 109:88–105, 2015.

[81] Thanh Hong Nguyen, Rong Yang, Amos Azaria, Sarit Kraus, and Milind Tambe. Analyzing the effectiveness of adversary modeling in security games. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[82] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic behavior*, 35(1-2):166–196, 2001.

[83] Frauke Oldewurtel, Colin N Jones, and Manfred Morari. A tractable approximation of chance constrained stochastic mpc based on affine disturbance feedback. In *IEEE Conference on Decision and Control*, pages 4731–4736, 2008.

[84] Christos G Panayiotou, William C Howell, and Michael Fu. Online traffic light control through gradient estimation using stochastic fluid models. *IFAC Proceedings Volumes*, 38(1):90–95, 2005.

[85] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[86] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[87] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.

[88] AC Pigou. The economics of welfare, 1920. *McMillan&Co., London*, 1932.

[89] James Pita, Richard John, Rajiv Maheswaran, Milind Tambe, Rong Yang, and Sarit Kraus. A robust approach to addressing human adversaries in security games.

In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1297–1298. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[90] Isaac Porche, Meera Sampath, R Sengupta, Y-L Chen, and Stephane Lafortune. A decentralized scheme for real-time optimization of traffic signals. In *Control Applications, 1996., Proceedings of the 1996 IEEE International Conference on*, pages 582–589. IEEE, 1996.

[91] Massimiliano Rak, Antonio Cuomo, and Umberto Villano. Mjades: Concurrent simulation in the cloud. In *Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on*, pages 853–860. IEEE, 2012.

[92] N Lawrence Ricker. Decentralized control of the tennessee eastman challenge process. *Journal of Process Control*, 6(4):205–221, 1996.

[93] Jeffrey S Rosenschein and Gilad Zlotkin. *Rules of encounter: designing conventions for automated negotiation among computers*. MIT press, 1994.

[94] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015. URL http://networkrepository.com.

[95] T. Roughgarden and E. Tardos. How bad is selfish routing? *Proceedings 41st Annual Symposium on Foundations of Computer Science*, 49(2):1–26, 2000. ISSN 0272-5428. doi: 10.1109/SFCS.2000.892069.

[96] Wenjie Ruan, Min Wu, Youcheng Sun, Xiaowei Huang, Daniel Kroening, and Marta Kwiatkowska. Global robustness evaluation of deep neural networks with provable guarantees for the l_0 norm. *arXiv preprint arXiv:1804.05805*, 2018.

[97] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

[98] Aaron Schlenker, Haifeng Xu, Mina Guirguis, Christopher Kiekintveld, Arunesh Sinha, Milind Tambe, Solomon Y Sonya, Darryl Balderas, and Noah Dunstatter. Don't bury your head in warnings: A game-theoretic approach for intelligent allocation of cyber-security alerts. In *IJCAI*, pages 381–387, 2017.

[99] David Schmeidler. Equilibrium points of nonatomic games. *Journal of Statistical Physics*, 7(4):295–300, 1973. ISSN 0022-4715. doi: 10.1007/BF01014905.

[100] Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for l 1-regularized loss minimization. *The Journal of Machine Learning Research*, 12:1865–1892, 2011.

[101] Manohar Shamaiah, Siddhartha Banerjee, and Haris Vikalo. Greedy sensor selection: Leveraging submodularity. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 2572–2577. IEEE, 2010.

[102] Shashank Shekhar, Hamzah Abdel-Aziz, Michael Walker, Faruk Caglar, Aniruddha Gokhale, and Xenofon Koutsoukos. A simulation as a service cloud middleware. *Annals of Telecommunications*, 71(3):93–108, 2016. doi: 10.1007/s12243-015-0475-6. URL http://dx.doi.org/10.1007/s12243-015-0475-6.

[103] Dan Shen, Genshe Chen, Jose B Cruz, and Erik Blasch. A game theoretic data fusion aided path planning approach for cooperative UAV ISR. In *Aerospace Conference*, pages 1–9, 2008.

[104] Yogesh Simmhan, Saima Aman, Alok Kumbhare, Rongyang Liu, Sam Stevens, Qunzhi Zhou, and Viktor Prasanna. Cloud-based software platform for big data analytics in smart grids. *Computing in Science & Engineering*, 15(4):38, 2013.

[105] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[106] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 2019.

[107] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[108] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013. URL http://arxiv.org/abs/1312.6199.

[109] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2013.

[110] Thomas L Thorpe. Vehicle traffic light control using sarsa. In *Online]. Available: citeseer. ist. psu. edu/thorpe97vehicle. html*. Citeseer, 1997.

[111] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[112] David I Urbina, Jairo A Giraldo, Alvaro A Cardenas, Nils Ole Tippenhauer, Junia Valente, Mustafa Faisal, Justin Ruths, Richard Candell, and Henrik Sandberg. 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. ACM, 2016.

[113] Heinrich Von Stackelberg. *Marktform und gleichgewicht*. J. springer, 1934.

[114] Bernhard Von Stengel and Shmuel Zamir. Leadership with commitment to mixed strategies. Technical report, Citeseer, 2004.

[115] Yevgeniy Vorobeychik and Murat Kantarcioglu. *Adversarial Machine Learning*. Morgan and Claypool, 2018.

[116] John Glen Wardrop. Road paper. some theoretical aspects of road traffic research. *Proceedings of the institution of civil engineers*, 1(3):325–362, 1952.

[117] Christopher K Williams and Carl Edward Rasmussen. Gaussian processes for machine learning. *the MIT Press*, 2(3):4, 2006.

[118] Zhennan Yan, Yiqiang Zhan, Zhigang Peng, Shu Liao, Yoshihisa Shinagawa, Shaoting Zhang, Dimitris N Metaxas, and Xiang Sean Zhou. Multi-instance deep learning: Discover discriminative local anatomies for bodypart recognition. *IEEE transactions on medical imaging*, 35(5):1332–1343, 2016.

[119] Zhengyu Yin, Manish Jain, Milind Tambe, and Fernando Ordónez. Risk-averse strategies for security games with execution and observational uncertainty. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[120] X-H Yu and Wilfred W Recker. Stochastic adaptive control model for traffic signal systems. *Transportation Research Part C: Emerging Technologies*, 14(4):263–282, 2006.

[121] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 103–117. ACM, 2017.

[122] Mengchen Zhao, Bo An, and Christopher Kiekintveld. Optimizing personalized email filtering thresholds to mitigate sequential spear phishing attacks. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[123] Yan Zhou, Murat Kantarcioglu, and Bowei Xi. A survey of game theoretic approach for adversarial machine learning. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3):e1259, 2019.

[124] Guus Zoutendijk. *Methods of feasible directions: a study in linear and non-linear programming*. Elsevier, 1960.