Secure Control for Autonomous Cyber-Physical Systems Under Temporal Logic Constraints

by

Luyao Niu

Submitted to the Electrical and Computer Engineering Department
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at

WORCESTER POLYTECHNIC INSTITUTE

April 2022

Certified by .....
Dr. Andrew Clark, Dissertation Committee Chair
Associate Professor, Electrical and Computer Engineering, WPI

Certified by .....
Dr. Raghvendra V. Cowlagi, Dissertation Committee
Associate Professor, Aerospace Engineering, WPI

Certified by .....
Dr. Jie Fu, Dissertation Committee
Assistant Professor, Electrical and Computer Engineering, UF

Certified by .....
Dr. Ziming Zhang, Dissertation Committee
Assistant Professor, Electrical and Computer Engineering, WPI

Secure Control for Autonomous Cyber-Physical Systems Under Temporal Logic Constraints

by

Luyao Niu

Submitted to the Electrical and Computer Engineering Department
on April 28, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Modern cyber-physical systems are expected to perform increasingly complex tasks. Synthesizing controllers to satisfy the assigned complex tasks with provable guarantees has received extensive research attention. The prior works assume that the cyber-physical systems are operated in benign environments where no adversary perturbs the system behaviors. Cyber-physical systems, however, have been demonstrated to be attractive targets of malicious attacks in a large variety of applications. To this end, the controllers designed for cyber-physical systems need to be capable of satisfying the complex specifications in the presence of malicious adversaries.

In this thesis, we consider cyber-physical systems operated in adversarial environments. The systems aim at synthesizing controllers so as to satisfy some complex specifications modeled using temporal logic formulas. We identify a sequence of important problem settings, and propose a solution to each of them. We first study the problem of maximizing the probability of satisfying a linear temporal logic specification, i.e., maximizing the satisfaction probability. There exists an adversary that tampers with the input of the actuator, aiming to minimize the satisfaction probability. We formulate the interaction between the system and adversary as a stochastic game, and develop a value iteration algorithm to synthesize the controller. We then extend the problem to the scenarios where the systems are given multiple temporal logic specifications that may not be satisfied simultaneously. We develop efficient algorithms to synthesize controllers that yield minimum violations of the specifications.

Next, we study the problem of control synthesis for cyber-physical systems with partial observabilities under linear temporal logic constraints. We formulate the interaction between the system and adversary as a partially observable stochastic game. We synthesize a controller equipped with finite memory for the system. We further investigate the problem of satisfying time sensitive specifications modeled using metric interval temporal logic for cyber-physical systems in the presence of attack. Incorporating the notion of time provides the adversary an additional attack surface, i.e., timing attack. We propose a durational stochastic game to capture the interaction between the system and the adversary that launches actuator attacks as well as timing attacks. We synthesize a controller for the

system to maximize the satisfaction probability.

Computing the aforementioned stochastic games is computationally demanding. In addition, synthesizing controllers using the stochastic games suffers from the curse of dimensionality for systems of large scales. To mitigate the computations, we study the problem of abstraction-free control synthesis for cyber-physical systems under linear temporal logic specifications. The abstraction-free synthesis eliminates the procedure of abstracting the cyber-physical system as a discrete finite abstraction. We decompose the linear temporal logic specification into a sequence of sub-formulas, and use a set of control barrier function based constraints to guarantee the satisfaction of each sub-formula.

Thesis Supervisor: Dr. Andrew Clark
Associate Professor, Electrical and Computer Engineering, WPI

Acknowledgments

First and foremost, I would like to offer my sincerest gratitude to my research advisor, Professor Andrew Clark, for his mentoring and advice during my Ph.D. study. It was my great hornor to have the chance of joining Secure Cyber-Physical Systems Lab and working with Professor Andrew Clark. Professor Clark has demonstrated and taught me how to conduct research and deliver the results. Professor Clark has also exemplified how hard work can turn into excitement and great joy over the journey. In addition, Professor Clark has generously shared his experience to relieve my anxiety, confusion, and hesitation over the past five years. I also would like to thank the service, comments, and helpful discussion of committee members Dr. Raghvendra V. Cowlagi, Dr. Jie Fu, Dr. Kaveh Pahalavan, and Dr. Ziming Zhang.

I would like to express my thanks to Professor Radha Poovendran of University of Washington for his guidance and advice. His experience and professionalism have demonstrated me how to become an impactful researcher. I extend my deep thanks to Professor Kaveh Pahlavan. The life experience shared by Professor Pahlavan helped my to hold the composure when I was faced with challenges.

It was my great fortune to have many excellent collaborators. I thank Dr. Jie Fu associated with University of Florida for her comments and advice on minimum violation control synthesis. I thank Dr. Bhaskar Ramasubramanian for his efforts in our joint work on control synthesis under partially observable environments, privacy-preserving resilience, and reinforcement learning beyond expectation. I thank Dinuka Sahabandu for our collaborations on multi-agent motion planning and controlled islanding for power systems. I also thank Zhouchi Li for his effort towards our joint work on reference tracking with safety and reachability guarantees under false data injection attacks. I would like to thank Hongchao Zhang for our collaboration on safety-critical control synthesis for sampled-data systems.

I extend my thanks to my labmates, Dr. Qiqiang Hou, Zhouchi Li, and Hongchao Zhang, for creating a warm environment for Secure Cyber-Physical Systems Lab and offering the helpful technical discussions during my Ph.D. study.

I would like to acknowledge the generous funding resources I have received during my Ph.D. study: National Science Foundation (NSF) grants CNS-1656981, CNS-1544173, and CNS-1941670, the Office of Naval Research (ONR) grants N00014-17-S-B001 and N00014-17-1-2946, and AFOSR grant FA9550-20-1-0074.

Finally, I would like to dedicate this work to my dear parents. Words cannot express my gratefulness for their selflessness love and support over the years, which made it possible for me to reach this stage of my life.

Contents

1 Introduction **1**
- 1.1 Maximizing Satisfaction Probability . . . 3
- 1.2 Minimizing Invariant Constraint Violation Rate . . . 3
- 1.3 Minimum Violation Control Synthesis . . . 4
- 1.4 Maximizing the Satisfaction Probability with Partial Observability . . . 5
- 1.5 Control Synthesis under Time-Critical Constraints . . . 5
- 1.6 Abstraction-Free Control Synthesis . . . 6
- 1.7 Organization of this Thesis . . . 6

2 Related Work **9**
- 2.1 Control Synthesis under Temporal Logic Constraints in the Absence of Adversaries . . . 9
- 2.2 CPS Security and Reactive Synthesis . . . 10
- 2.3 Partial Satisfaction of Temporal Logic Constraints . . . 11
- 2.4 Control Synthesis under Partial Observabilities . . . 11
- 2.5 Control Synthesis under Time-Critical Specifications . . . 12

3 Background **15**
- 3.1 Markov Decision Processes . . . 15
- 3.2 Classical Problems and Algorithms on Markov Decision Processes . . . 16
  - 3.2.1 Discounted Total Reward Maximization Problem . . . 17
  - 3.2.2 Average Reward per Stage Problem . . . 18
  - 3.2.3 Average Reward per Cycle Problem . . . 19
- 3.3 Stochastic Games . . . 20
- 3.4 Formal Specifications: Temporal Logics and Automata . . . 21

4 Optimal Secure Control under LTL Constraints **27**
- 4.1 Introduction . . . 27
- 4.2 Sysem Model and Problem Statement . . . 28
- 4.3 Control Synthesis to Maximize Satisfaction Probability . . . 30
  - 4.3.1 Computation of the Probability of Satisfying LTL Specification φ . . . 30

4.3.2 Equivalence to the Problem of Maximizing Reachability Probability . 32
4.3.3 Value Iteration Algorithm for Control Synthesis 36
4.4 Case Study 38
4.5 Conclusion 39

5 Secure Control under LTL Constraints with Minimal Invariant Constraint Violation Rate 43
5.1 Introduction 43
5.2 Sysem Model and Problem Statement 44
5.3 Control Synthesis to Minimize Invariant Constraint Violation Rate 45
5.3.1 Optimality Condition Derivation 46
5.3.2 Policy Iteration Algorithm for Control Synthesis 51
5.4 Case Study 54
5.5 Conclusion 56

6 Optimal Minimum Violation Control under Multiple LTL Constraints 57
6.1 Introduction 57
6.2 System Model and Problem Statement 58
6.3 Proposed Approach for Minimum Violation Control Synthesis 60
6.3.1 Product SG Construction 61
6.3.2 A Mixed Integer Non-linear Programming Formulation 64
6.3.3 Exact Algorithm for Problem 6.1 66
6.3.4 Approximate Algorithm for Problem 6.1 71
6.4 Case Study 75
6.5 Conclusion 80

7 Optimal Secure Control under LTL Constraints in Partially Observable Environments 81
7.1 Introduction 81
7.2 System Model and Problem Formulation 82
7.3 Control Synthesis to Maximize Satisfaction Probability with Partial Observability 84
7.3.1 Product POSG and Recurrent Sets 84
7.3.2 Determining Candidate FSCs of Fixed Sizes 87
7.3.3 Value Iteration for POSGs 90
7.3.4 Varying the Size of Controller's FSC 94
7.4 Case Study 99
7.5 Conclusion 103

8 Control Synthesis under Time-Critical Constraints in the Presence of Timing and Actuator Attacks 105
8.1 Introduction . . . 105
8.2 System Model and Problem Statement . . . 106
8.2.1 Adversary and Controller Models . . . 107
8.2.2 Durational Stochastic Game . . . 107
8.2.3 Problem Statement . . . 108
8.3 Control Synthesis under MITL Constraints in the Presence of Timing and Actuator Attacks . . . 110
8.3.1 Product Durational Stochastic Game Construction . . . 110
8.3.2 Controller Policy Representation: Finite State Controllers . . . 111
8.3.3 Value Iteration Algorithm for Control Synthesis . . . 112
8.4 Case Study . . . 118
8.4.1 Signalized Traffic Network Model . . . 118
8.4.2 Two-Tank System . . . 122
8.5 Conclusion . . . 124

9 Abstraction-Free Control Synthesis under Temporal Logic Constraints 125
9.1 Introduction . . . 125
9.2 Preliminary Background . . . 126
9.3 System Model and Problem Formulation . . . 128
9.4 Abstraction-Free Control Synthesis using Control Barrier Functions . . . 129
9.4.1 Time Varying ZCBF and FCBF . . . 129
9.4.2 Design of Control Barrier Functions . . . 130
9.4.3 CBF-based Controller Synthesis . . . 135
9.5 Case Study . . . 137
9.6 Conclusion . . . 139

10 Conclusions and Future Work 141

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

1-1 Organization of this thesis. 2

3-1 An example of MDP modeling a robot navigating in a 2×3 grid world. Each grid is a state of the MDP. The robot has four actions, denoted as $U = \{L, R, U, D\}$, representing 'moving left', 'moving right', 'moving up', and 'moving down'. 16

3-2 An example of MDP modeling a robot navigating in a 2×3 grid world. Each grid is a state of the MDP. The robot and the adversary each has four actions, denoted as $U_1 = U_2 = \{L, R, U, D\}$, representing 'moving left', 'moving right', 'moving up', and 'moving down'. 20

3-3 The DRA representing LTL formula $\varphi = \Diamond(a \wedge \Diamond b)$. The states are represented by circles and the transitions are shown by arrows. 23

4-1 Comparison of the proposed approach and the approach without considering the presence of the adversary. Fig. 4-1a gives the trajectories obtained using two approaches. The solid blue line is the trajectory obtained using the proposed approach, while the dashed red line represents the trajectory obtained using the approach without considering the presence of the adversary. Fig. 4-1b and Fig. 4-1c present the probability of satisfying the LTL specification using the proposed approach and the approach without considering the adversary when the initial state is set as each of the states lands in the intersections of the grid world, respectively. The shade of gray level at the intersection states corresponds to the satisfaction probability, with black being probability 0 and white being probability 1. 41

5-1 Comparison of the proposed approach and the approach without considering the presence of the adversary. Fig. 5-1a gives the trajectories obtained using two approaches. The solid blue line is the trajectory obtained using the proposed approach, while the dashed red line represents the trajectory obtained using the approach without considering the presence of the adversary. Fig. 5-1b shows the expected invariant constraint violation cost with respect to iteration indices. 55

6-1 Fig. 6-1a presents the grid world that the patrol unit and intruder interact within. The initial locations of the patrol unit and intruder are $s_{1,1}$ and $s_{3,2}$, respectively. The target grid is $s_{2,3}$, and the obstacle region is $s_{3,3}$. Fig. 6-1b shows the illustration of the stochastic game. The row index gives the grid that the patrol unit is located in, while the column index gives the grid that the intruder is located in. The cells colored in black with row index 9 are the obstacles that the patrol unit needs to avoid, and the cells colored in black with column index 9 are the obstacles that the intruder needs to avoid. The cells located on the minor diagonal are the ones that models the event that the patrol unit captures the intruder, i.e., the locations of the patrol unit and intruder coincide. 75

6-2 The complete DFAs associated with specification ϕ_1 and ϕ_3. Fig.6-2a is the DFA for specification $\phi_1 = \neg obstacle$ U $target$. Fig. 6-2b is the DFA for specification $\phi_2 = \diamond capture\_intruder \vee \diamond adversary\_avoid\_obstacle$. 76

6-3 Fig. 6-3a shows the expected reward that the patrol unit can obtain by committing to the control policy synthesized following Algorithm 9 at each iteration when the initial locations of the patrol unit and intruder are $s_{1,1}$ and $s_{3,1}$, respectively. Fig. 6-3b shows the expected reward for the patrol unit by committing to the control policy synthesized following Algorithm 10 at each iteration when the initial locations of the patrol unit and intruder are $s_{1,1}$ and $s_{3,1}$, respectively. 76

6-4 Fig. 6-4a compares the expected reward collected by the patrol unit using different approaches. Algorithm 9 achieves the highest expected reward. Algorithm 10 gives a sub-optimal solution. Baseline 1 and 2 achieves less expected reward compared with the proposed approaches. Fig. 6-4b gives the the expected reward collected by the patrol unit using Algorithm 9 and Algorithm 10 under different λ used in anchoring bias model (6.1). The value of λ varies from 0 to 1 with step size 0.1. 79

7-1 Value function of a two-state POSG with three states in $\mathcal{C}_C$. The value of each FSC state is linear in the belief (black lines). The value function is the point-wise maximum of the values of the FSC states (red curve). 95

7-2 Robust linear program for state 2 of $\mathcal{C}_C$. Improved vector $V'^2_{g_C} + \epsilon$ is tangent to the one-step look-ahead value function. 95

7-3 At a local equilibrium, all states are tangent to the one-step look-ahead value function. 97

7-4 Clockwise, from *top-left*: Global Markov chain (GMC) for initial controller and adversary FSC structures- green states (m_1&m_2) must be visited infinitely often, and state in red (m_3) must be visited finitely often in steady-state; GMC state $m_i \in S \times Q \times G_C \times G_A$; State-space for $M = 3, N = 2$ showing unsafe (s_4) and target (s_5) states. 99

7-5 The agent aims to satisfy the LTL formula $\varphi = \square\diamond\texttt{tar} \wedge \square\neg\texttt{obs}$ in the presence of an adversary, in a partially observable environment. The environment is the grid-world in Fig.7-5a. The states in red indicate the presence of an obstacle, the state in green is the target state, and the agent starts in state s_0. The agents' actions are determined by their observations of the state. Assume that the controller FSC has at least as many states as the adversary FSC, i.e. $|G_C| \geq |G_A|$. Fig. 7-5b shows the fraction of runs (out of 100) when the agent reaches the target within 80 steps and 40 steps. This number is higher when $|G_C|$ is larger, for a fixed value of $|G_A|$ ($-\texttt{o}-$, $-*-$, and $-\diamond-$ curves). For the same $|G_C|$, the fraction of successful runs is higher when $|G_A|$ is lower ($-\texttt{o}-$ and $-*-$ curves). The fraction of successful runs is also higher when the agent is allowed more steps to reach the target ($-\texttt{o}-$ and $-\diamond-$ curves). 100

8-1 Representation of a *signalized traffic network*. The network consists of 4 intersections and 16 links. Intersections are represented by squares, and links by arrows. Dotted arrows denote outgoing links that are not explicitly modeled. 119

8-2 Number of vehicles on Links 2, 3, and 4 at each time corresponding to the MITL formula $\varphi_3 = \diamond_{[0,5]}((x_2 \leq 10) \wedge (x_3 \leq 10) \wedge (x_4 \leq 10))$. In the presence of an adversary, the controller adopts an FSC-based policy with one realization shown in Table 8.1. The dotted horizontal line is the threshold for the maximum number of vehicles allowed (= 10). The three curves indicate that the number of vehicles in the links satisfies the MITL objective since they are each lower than 10 before 5 time units. 121

8-3 Evaluation on a two-tank system for an MITL specification that requires water levels in the tanks to be at least 0.3, and to be within 0.1 of each other, before time $k = 5$. An FSC-based controller policy is compared with a baseline policy that does not account for the presence of an adversary. Fig. 8-3a shows the water level in the second tank, using the two policies. The solid line represents the FSC-based policy, while the dashed and dash-dot lines represent the baseline in the presence and absence of the adversary, respectively. The absolute value of the difference between water levels in the two tanks using the two policies is presented in Fig. 8-3b. The solid line with circle markers represents the FSC-based policy, while the dashed line and dash-dot line represent the baseline policy under adversarial and benign environments, respectively. We observe that the baseline policy satisfies the MITL objective in the absence of the adversary, but fails to do so when an adversary is present. The FSC-based policy, in contrast, satisfies the MITL objective in the presence of the adversary. 123

9-1 Fig. 9-1a shows the DRA associated with $\varphi = \diamond A \wedge \diamond B \wedge \square(\neg O \wedge C)$. It has one Rabin pair $(\emptyset, \{q_3\})$. Fig. 9-1b presents the trajectories of both robot using the proposed approach in this chapter. The trajectory of the first robot is plotted in solid line, while the trajectory of the second robot is plotted in dotted line. The first robot eventually reaches region A, and the second robot eventually reaches region B. Both robots successfully avoid the obstacle region O and remain close enough to each other. CBFs h_{ψ_1} and h_{ψ_2} are presented in Fig. 9-1c. h_{ψ_1} is plotted in dotted line and h_{ψ_2} is plotted in solid line. Both CBFs become positive at some time $t \geq 0$, at which the formulas corresponding to the CBFs are satisfied. 137

List of Tables

4.1 Comparison of probabilities of satisfying specification φ when starting from the states located in intersections using proposed approach and approach without considering the adversary. 39

7.1 Satisfaction probability and standard deviation (over 100 trials) of reaching the target state s_{19} from s_0 for LTL formula $\varphi = \square\diamond\texttt{tar} \wedge \square\neg\texttt{obs}$ starting from s_0 for varying number of defender FSC states $|G_C|$, when number of states in adversary FSC, $|G_A| = 1$ and $|G_A| = 2$. 102

7.2 Comparison of probabilities of satisfying LTL formula $\varphi = \square\diamond\texttt{tar} \wedge \square\neg\texttt{obs}$ starting from s_0 in the presence and absence of an adversary. The first column lists the number of defender FSC states. Subsequent columns enumerate satisfaction probabilities in the following scenarios: i) absence of adversary (Benign Baseline [1]); ii) using a defender policy that was synthesized without an adversary, but realized in the presence of an adversary (Adversarial Baseline); iii) using a defender policy designed assuming the presence of an adversary (Adversary-Aware Design- **our approach**). Although the benign baseline gives the highest satisfaction probability, the same baseline when used in the presence of an adversary results in a much lower satisfaction probability. In comparison, our *Adversary-Aware Design* approach results in a higher satisfaction probability than the 'Adversarial Baseline' where we use a defender policy designed to account for adversarial behavior. We assume that the adversary FSC has one state, so that the GMC with and without the adversary FSC will have the same number of states. 102

8.1 A sample sequence of the traffic light realized at each intersection for the MITL specification $\varphi_3 = \diamond_{[0,5]}\left((x_2 \leq 10) \wedge (x_3 \leq 10) \wedge (x_4 \leq 10)\right)$. The letter 'R' represents a 'red' signal, and 'G' represents 'green' signal. 120

Chapter 1

Introduction

Cyber-physical systems (CPS) tightly integrate sensing, communication, computation, and control [2]. CPS are already having transformative impact in applications including energy systems [3], medical devices [4], robotics and smart transportation [5].

Due to the tight coupling between cyber and physical components, CPS are exposed to new threats raised by malicious cyber attacks, which have been reported in multiple CPS domains, including power systems [6], automobiles [7, 8, 9], surgical robots [10], and nuclear reactors [11]. These attacks can modify the observations perceived by CPS, bias the decisions, and manipulate the system behaviors. For instance, a malicious adversary can manipulate the sensor readings by spoofing the sensors [12, 13, 14] and hijacking the communication channel between the sensor and controller [15]. The control commands issued by the controller can be manipulated by an adversary via false data injection attacks [16], and can be jammed by an adversary via denial-of-service attacks [17].

CPS are additionally expected to perform increasingly complex tasks. To specify the desired tasks and properties of CPS, we need an expressive, concrete, and rigorous 'language'. To this end, formal logic specifications modeled using linear temporal logic (LTL) and computation tree logic (CTL) are widely adopted [18, 19]. Typical examples of tasks that are encoded using temporal logics include liveness (e.g., "always eventually A"), safety (e.g., "always not A"), and priority (e.g., "first A, then B") [18]. For systems operated in stochastic environments or imposed probabilistic requirements (e.g., "reach A with probability 0.9"), probabilistic extensions have also been proposed [20]. Autonomous control synthesis with verifiable guarantees on the satisfaction of given specifications motivates the concept of *correct-by-construction* control synthesis, which has received increasing attention.

The aforementioned challenges faced by CPS, namely the presence of malicious adversaries and correct-by-construction control synthesis, have been addressed separately at different levels. The security of CPS is mostly studied focusing on designing feedback control laws and state estimations, without considering any complex system specification. The control synthesis under complex specifications normally focuses on the logical decision-making controllers, assuming there exists no adversary that tampers with the behavior of the sys-

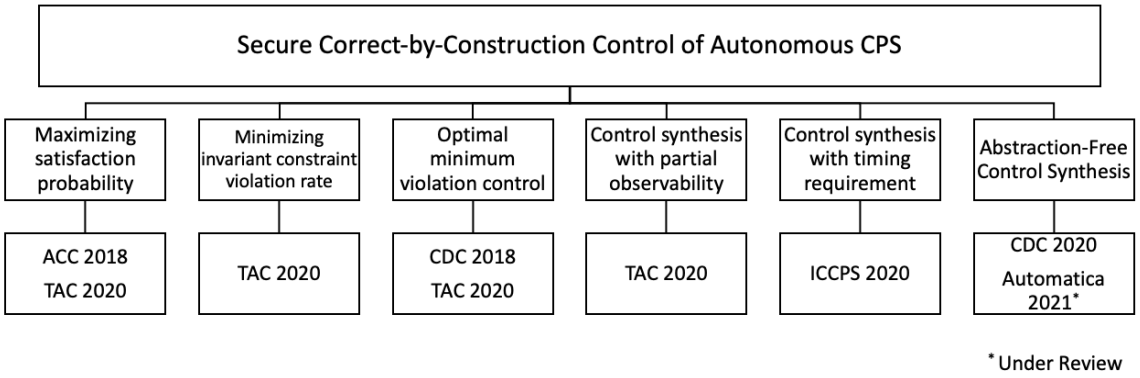


Figure 1-1: Organization of this thesis.

tem. The main objective of the logical decision-making controllers is to satisfy the given tasks by computing discrete and coarse control actions for CPS, which will be used to guide the calculation of continuous control inputs from the motion-planning controllers. We detail how each challenge is resolved as follows:

- Defenses and protections of CPS against cyber attacks have been extensively studied using control- and game-theoretic approaches [21, 22, 12, 23, 24, 25]. These approaches merely focus on detecting and isolating the attacks so as to continue perform the task, e.g., minimizing a quadratic cost. When the CPS are assigned more complex tasks, such approaches may not be applicable to provide any verifiable guarantee on the satisfactions of the tasks.

- Multiple frameworks have been proposed to synthesize controllers for CPS operated under benign environments [26, 27, 28, 29, 30]. Although these approaches are able to consider the non-determinism and probabilistic behaviors of the CPS, they may become suboptimal or invalid when applied to CPS in the presence of malicious adversaries. The reasons are two-fold. First, unlike stochastic errors or modeling uncertainties, intelligent adversaries are able to adapt their strategies to maximize their impacts against a given controller, and thus exhibit strategic behaviors. Moreover, CPS will have limited information regarding the objectives and strategies of the adversaries, making techniques such as randomized control strategies potentially effective in mitigating attacks.

To the best of our knowledge, however, automatic and secure correct-by-construction control synthesis for CPS subject to malicious attacks has received limited research attention. It is this gap that this thesis aims to fill. To this end, we model the adversary as an additional decision making agent, and formulate the interaction between CPS and adversary using game theory. We identify a sequence of important problem settings, as shown in Fig. 1-1. We

formulate each problem, and propose one or multiple algorithms to solve each of them. We show that the algorithms developed in this thesis are sound and converge within finitely many iterations. The specific contributions are described as follows.

1.1 Maximizing Satisfaction Probability

Due to the presence of malicious adversaries, it might be overly conservative and restrictive to consider deterministic or almost-sure satisfaction of the given temporal logic specification. To this end, one may only be able to compute a control strategy such that the worst-case probability of satisfying the given specification is maximized, where the worst-case scenario models the fact that the adversary can intelligently act to minimize such probability.

We denote the probability of satisfying the specification as satisfaction probability, and investigate the problem of control synthesis for CPS to maximize the worst-case satisfaction probability of an LTL formula. We first propose a heuristic algorithm to abstract the CPS in the presence of a malicious adversary as a stochastic game. We then prove that the worst-case satisfaction probability is equivalent to the worst-case reachability probability to a subset of states on the stochastic game. Such subset of states are those starting from which we can guarantee almost-sure satisfaction of the specification, regardless of the strategy committed by the adversary. We show that such set of states can be identified by verifying the connectivities of the underlying graph of the stochastic game. We prove that the max-min reachability probability admits a fixed point, and develop a value iteration algorithm to compute the max-min reachability probability. Our case study implies that the control policy synthesized using the proposed approach significantly outperforms the one obtained using the existing approaches without considering the presence of the adversary.

1.2 Minimizing Invariant Constraint Violation Rate

As CPS are expected to perform increasingly complicated tasks, the specifications given to CPS become more and more complex. Such complex specifications may overly constrain the system behavior, leading to zero satisfaction probability and making the control synthesis with maximal satisfaction probability infeasible. In this case, one may allow partial violation on the specification so that the control synthesis becomes feasible. This problem can be important and brutal in some scenarios. For example, we may have to synthesize a controller to stablize an unstable CPS, i.e., satisfying a stability specification, at the cost of violating some performance constraint.

We formulate a stochastic game which captures the interaction between the controller of a CPS and an adversary. We consider the case where the LTL specification given to the CPS is infeasible, i.e., satisfaction probability zero. We consider a class of LTL specifications that consists of an arbitrary LTL formula φ_1 and an invariant property, where an invariant property is one defined on each state of the stochastic game. We allow violations on the

invariant property so as to obtain a feasible controller that can partially satisfies the original LTL formula, i.e., satisfy φ_1. The goal is then to synthesize a controller that (i) maximizes the satisfaction probability of φ_1, and (ii) minimizes the cost incurred due to violating the invariant property.

The metric we use to measure the impact of violating the invariant constraint is the average cost per cycle, where a cycle is defined as the time period between two consecutive satisfactions of LTL specification φ_1. We solve the problem by bridging it with a problem named average cost per stage problem studied on Markov decision processes. We derive the optimality condition for minimizing invariant constraint violation rate, and develop a policy iteration algorithm to synthesize a control policy such that the cost incurred due to violating invariant constraint per cycle is minimized.

1.3 Minimum Violation Control Synthesis

In some applications, CPS are required to compute a control policy to satisfy multiple specifications. However, due to the incompatibility among these specifications and/or the presence of the adversary, these specifications cannot be satisfied simultaneously. For instance, the specifications requiring the system to always stay within regions A and B while A and B have empty intersection cannot be satisfied at the same time. This motivates the need for minimum violation control synthesis for CPS.

We consider a stochastic game that is given a set of specifications modeled using LTL co-safe. We assign a reward to each specification modeling its priority. We then formulate the minimum violation control synthesis problem by maximizing the expected reward that can be achieved by satisfying the specifications. We additionally consider the limited observation capability from the adversary, capturing the practical scenarios where human adversaries normally cannot make perfect observations or are biased by their beliefs. We show that the minimum violation control policy can be calculated by solving a mixed integer non-linear program (MINLP). We propose two algorithms to solve the MINLP. The first algorithm adaptively updates a reference distribution so as to approximate the optimal solution to the MINLP. This algorithm converges to the optimal solution with probability one, without any guarantee on the convergence rate. To overcome the potential slow convergence incurred by this algorithm, we propose a heuristic algorithm as an approximate solution. The heuristic algorithm converges to a local optimal solution within a certain number of iterations. Our numerical evaluation suggests that the heuristic algorithm achieves near-optimal performance.

1.4 Maximizing the Satisfaction Probability with Partial Observability

CPS may not always be capable of fully observing their state information. For example, it may be economically or physically infeasible to equip the system with a multitude of sensors. In these cases, CPS can only have partial observabilities. Control synthesis under partial observabilities is different from that with full observations since the state information is not directly known to either the CPS or the adversaries. Thus the CPS and adversaries need to make their decisions based on their observations. It has been reported that determining the optimal control policy for CPS with partial observabilities in the *absence* of the adversary is NP-hard [31].

We formulate the problem of synthesizing a control policy that maximizes the satisfaction probability for CPS with partial observabilities in the presence of adversaries. We formulate the CPS with partial observabilities in the presence of adversaries using partially observable stochastic games (POSGs). We represent the policies under partial observations using finite state controller, and equivalently convert the problem of maximizing satisfaction probability on POSGs to the problem of maximizing reachability probability. This conversion allows us to develop a value iteration algorithm to compute finite state controllers for CPS as approximate solutions. We also show that increasing the size of the finite state controllers can improve the satisfaction probability, and present an algorithm to do so.

1.5 Control Synthesis under Time-Critical Constraints

Although LTL specifications specify the orders of the occurrence of the events, they are not suitable to specify tasks for CPS involving deadlines or time intervals. When the CPS are expected to perform time-critical specifications, other temporal logic specifications such as metric interval temporal logic (MITL) can be used.

When the CPS is given a time-critical specification, we need to focus on the *timed system behavior*. As a consequence, the adversary can exploit an additional surface to attack the system and bias the system behavior, i.e., the adversary can either deviate the system trajectory to some undesired state, or manipulate the time instance that certain properties become true, to violate the time-critical specification, which makes control synthesis for CPS with maximal satisfaction probability more challenging, compared with CPS under LTL constraints.

We investigate the problem of control synthesis to maximize the satisfaction probability of an MITL specification for CPS in the presence of adversaries. We consider the adversaries that have two capabilities: (i) tamper with the control input applied by the system via an actuator attack, and (ii) manipulate the timing information perceived by the system via a timing attack. We propose an entity named durational stochastic game (DSG) to capture the interaction between the CPS and adversary. DSGs are more powerful than stochastic games in the sense that they are capable of not only modeling the ordering of the events as

stochastic games are, but also the time consumption required for each event. We develop a value iteration algorithm to compute the max-min satisfaction probability on DSG. Our case study results indicate that the proposed approach outperforms the baselines where the adversary can cause zero satisfaction probability via actuator and timing attacks.

1.6 Abstraction-Free Control Synthesis

The hierarchical architecture for control synthesis of CPS grants us the advantage to lift from continuous domain to discrete domain via computing finite abstractions (e.g., stochastic games, POSGs, DSGs), so as to be better aligned with the semantics of temporal logic specifications, which are normally captured by finite state automata. We can then leverage formal methods in the context of model checking [18] to synthesize controllers to satisfy the given temporal logic specifications. The finite abstractions can be generated by partitioning the state space and control input space of CPS with consistency guarantees (e.g., simulation and bisimulation relations [32]). However, computing the abstractions can be computationally demanding and may suffer from the curse of dimensionality. This motivates the topic of abstraction-free control synthesis that aims at computing the control input for the system while avoiding generating these abstractions.

We investigate the problem of abstraction-free control synthesis for CPS under LTL specifications. Our solution approach leverages the paradigm of hybrid systems, where we use the discrete automaton representing the LTL specification to track the satisfaction of the specification, and use continuous system dynamics to constrain the system behavior. Given the automaton corresponding to the LTL specification, we decompose an accepting run on it into a sequence of sub-formulas. We then map the satisfaction of each sub-formula to reaching a subset of continuous system states. We construct a set of control barrier function constraints to guarantee that the system can reach the desired subset of continuous states, and formulate a quadratic program to synthesize the continuous controller. Although this problem has not incorporated the presence of any adversary, we believe this is an important initial step for developing abstraction-free control synthesis for CPS in the presence of adversaries.

1.7 Organization of this Thesis

This thesis is organized as follows. Chapter 2 introduces related work. Chapter 3 presents some preliminary background. Chapter 4 develops a framework for automatic control synthesis that maximizes the satisfaction probability. Chapter 5 gives the control synthesis that minimizes the invariant constraint violation rate. Chapter 6 discusses the minimum violation control synthesis under multiple LTL constraints. Control synthesis for systems with partial observations is investigated in Chapter 7. Chapter 8 presents the control synthesis for CPS under time-critical specifications in the presence of both actuator and timing

attacks. Abstraction-free control synthesis is studied in Chapter 9. Chapter 10 concludes the thesis and discusses some future work.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Related Work

This chapter presents an overview of the related work.

2.1 Control Synthesis under Temporal Logic Constraints in the Absence of Adversaries

Temporal logics such as LTL and CTL are widely used to specify and verify system properties [18], especially complex system behaviors. In this section, we review the literature on control synthesis to satisfy temporal logic specifications when there exists no adversary. We divide the related work into two categories: namely the abstraction-based approaches and abstraction-free approaches.

The abstraction-based approaches follow a hierarchical architecture. These approaches rely on computing a finite abstraction of the CPS model. Typical examples of finite abstractions include finite transition systems [29, 27, 33], tree structures [34, 35], Markov decision processes (MDPs) [26, 28, 36, 30]. In the meantime, the temporal logic specifications are expressed using finite automata. Then the finite abstraction representing the system model and the finite automaton modeling the temporal logic specification are integrated together. Applying off-the-shelf model checking algorithms on this integrated entity yields the higher level controllers with verifiable guarantee on the satisfaction of the temporal logic specification [18]. Finally, a lower level continuous controller can be computed by guaranteeing that (i) the requirements specified by the higher level controller are met, and (ii) the system dynamics are satisfied. These approaches, however, do not consider the presence of malicious adversaries that can tamper with the behaviors of the CPS. For CPS operated in adversarial environments, the aforementioned approaches may be suboptimal or invalid. In Chapter 4 to 8, we investigate how to synthesize the higher level controllers under different settings to satisfy the temporal logic constraints. We use stochastic games to better formulate the strategic interactions between the controller and adversary [37].

Abstraction-free approaches aim at mitigating the intense computation required when

constructing the finite abstractions of CPS. An optimization based approach is proposed in [38]. The authors encode the LTL constraints as mixed-integer linear constraints. The authors of [39] study control synthesis under co-safe LTL specifications. They compute the controller by solving a sequence of stochastic constrained reachability problems, which relies on solving nonlinear PDEs. The authors of [40] formulate a hybrid system using the automaton and the continuous system, and compute the controller by solving a mixed continuous-discrete HJB equation. Normally PDEs and HJB equations are approximately solved using computationally expensive numerical approaches such as Finite-Difference [41]. In Chapter 9 of this thesis, we give a different control barrier function (CBF) based approach to synthesize a controller by solving a quadratic program at each time.

The concept of CBF extends barrier function to CPS with control inputs [42]. Recent works have developed CBF based approaches for safety-critical systems [43, 44]. LTL specifications, however, capture a richer set of properties that cannot be modeled by safety constraints alone.

CBFs have gained popularity for CPS under LTL constraints [45, 46] and signal temporal logic (STL) constraints [47, 48]. The authors of [45, 46] investigate the problem of control synthesis of multi-robot system under LTL constraints using CBFs. Under their approaches, they solved a sequence of reachability problems using CBFs, and proposed a relaxation of CBF based constraints. Our approach differs from [45, 46] in the following two aspects. First, our approach considers a broader fragment of LTL compared to [45]. A fragment of LTL specification named $\text{LTL}_{Robotic}$ is considered in [45], while we consider LTL without next operator. Second, we propose a different approach to resolve the infeasibility between CBF constraints compared to [45, 46]. As we will demonstrate later, our proposed approach in Chapter 9 synthesizes a controller for a specification that is infeasible using the approaches in [45, 46], and thus serves as a complement to [45, 46].

2.2 CPS Security and Reactive Synthesis

CPS security has been investigated using game- and control- theoretic approaches [21, 22, 12, 23, 24, 25]. Secure state estimation is investigated in [21, 24, 25]. CPS security and privacy using game-theoretic approach is surveyed in [22]. LQG controller design under false data injection attacks is studied in [12]. Game theory based resilient control is considered in [23]. CPS security under Stackelberg setting and Nash setting are studied in [49] and [50], respectively. Stochastic Stackelberg security games have been studied in [51, 52] and references therein. These works mostly focus on the lower level controller design, e.g., LQG controller design, and thus are not readily applicable to systems that are subject to high level specifications such as temporal logic constraints.

Reactive synthesis under temporal logic constraints aims at automatic control synthesis to fulfill tasks in which the system behavior depends on the information gathered at runtime. In this case, the environment is treated as an adversary to capture the worst-case system behavior [19, 53, 54, 55, 56]. There are several aspects that distinguish this thesis

with the treatment in reactive synthesis [19, 53, 54, 55, 56]. First, turn-based games are normally considered in reactive synthesis, while concurrent game is considered in this thesis. In turn-based games, exactly one player is allowed to take action at each time step. Turn-based games are used to model the case where the players interact in asynchronous way. In concurrent games, all the players take actions simultaneously at each time step, and thus is more general compared with turn-based game. Moreover, reactive synthesis normally assumes the agent has full knowledge of the environment and information asymmetry between players cannot be captured. In this thesis, the Stackelberg setting is considered as shown in Chapter 4 to 8, which captures the hierarchical structure when players commit to strategies. Additionally, the approaches given in this thesis can be easily extended to handle the scenarios where the players have limited observation capabilities (see Chapter 6 and 7).

2.3 Partial Satisfaction of Temporal Logic Constraints

Failure of controller synthesis is first investigated in [57, 58]. The authors of [59] diagnose the unrealizable part of a specification, i.e., the part of specification that cannot be implemented by the system. Following [57, 58], minimum violation problem of deterministic systems is investigated in [60, 61, 62, 63, 64]. In [65], the problem of partial satisfaction of LTL safety specification on deterministic transition system is investigated. The authors of [65] divide the LTL specifications into hard and soft specifications, and solves the problem by proposing a quantitative semantics of LTL. A penalty structure is designed in [66] for motion planning under LTL constraint. Control synthesis for deterministic finite transition systems under infeasible temporal logic constraints is studied in [67, 68, 69, 70]. These works propose some metrics as measures to partial satisfaction of specifications, and solve the control synthesis problem as an optimization problem with respect to the proposed metrics. Another approach to address unrealizable specification is to repair the model [71, 72] – that is, introducing small changes to the model to make the original specification realizable.

In Chapter 5, we study the problem of control synthesis with minimum invariant property violation per cycle. In Chapter 6, we investigate the minimum violation control synthesis under multiple LTL constraints. In both chapters, we consider the presence of a malicious adversary who can adjust its strategy accordingly, which leads to an additional decision maker in the system. The presence of the adversary cannot be captured by any of the aforementioned work. Additionally, the frameworks that we will propose in Chapter 5 and 6 are applicable to stochastic systems, which are more general than deterministic transition systems.

2.4 Control Synthesis under Partial Observabilities

This section discusses related work on control synthesis under partial observabilities, in the contexts of partially observable MDPs (POMDPs) and partially observable stochastic

games (POSGs). A policy in a POSG (or POMDP) at time t depends on actions and observations at all previous times. A memoryless policy, on the other hand, only depends on the current state. Heuristics to approximately solve POMDPs include belief replanning [73], most likely belief state policy and entropy weighting [74], grid-based methods [75], and point-based methods [76]. The synthesis of parameterized finite state controllers (FSCs) for a POMDP to maximize the probability of satisfying of an LTL formula (in the absence of an adversary) is proposed in [77] and [1]. This is an approximate strategy since it does not use the observation and action histories; it uses only the most recent observation in order to determine an action. This restricts the class of policies that are searched over, but the finite cardinality of states in an FSC makes the problem computationally tractable. The authors of [78] show the existence of $\epsilon-$optimal FSCs for the average cost POMDP. Similar to MDP, a POMDP is only capable of representing the presence of one decision maker, i.e., the system controller, and thus is not sufficient to model the interaction between the controller and adversary.

Dynamic programming for POSGs for the finite horizon setting was studied in [79]. When agents cooperate to earn rewards, the framework is called a decentralized-POMDP (Dec-POMDP). The infinite horizon case for Dec-POMDPs is presented in [80], where the authors proposed a bounded policy iteration algorithm for policies represented as joint FSCs. A complete and optimal algorithm for deterministic FSC policies for DecPOMDPs is presented in [81]. Optimization techniques for 'fixed-size controllers' to solve Dec-POMDPs are investigated in [82]. A survey of recent research in Dec-POMDPs is presented in [83].

In Chapter 7, the problem of control synthesis to satisfy an LTL formula in partially observable adversarial environments is studied. FSCs are used to present the policies of the system and adversary. Chapter 7 presents a value iteration based procedure with convergence guarantee.

2.5 Control Synthesis under Time-Critical Specifications

We next discuss control synthesis for CPS under time-critical specifications. Time-critical specifications are commonly seen in real world applications such as real time scheduling problem [84, 85] and vehicle routing problem [86]. Time-critical tasks can be formulated using temporal logic specifications such as metric temporal logic (MTL), metric interval temporal logic (MITL), and signal temporal logic (STL), which not only specify the relative ordering of the events, but also the time duration between consecutive events.

Semi-Markov decision processes (SMDPs) [87] are used to model Markovian dynamics where the time taken for transitions between states is a random variable. SMDPs have been typically used to analyze problems in production scheduling [88, 89] and optimization of queues [90, 91, 92]. Similar to MDPs, SMDPs are not applicable to capture the interaction between two decision makers, i.e., the controller and adversary in the context of this thesis. The satisfaction of an MITL formula in a motion-planning context was studied in [86, 93, 94]. In [86], the authors propose a mixed-integer linear programming procedure that solved

the problem to optimality. The authors of [93] synthesize switching controllers for non-linear dynamical systems in order to satisfy MTL formulas. They present a correct-by-construction procedure which ensured that closed-loop trajectories of the system satisfied the MTL formulas. A timed automaton (TA) based approach to generate a plan for a robot to accomplish a task specified as an MITL formula is proposed in [94]. Control synthesis under STL constraints is studied in [47, 48]. However, the aforementioned works are tailored for a single agent, and do not consider the presence of an adversary. Moreover, the analysis in [86, 93, 94] restricts their focus to MITL formulas with reachability accepting conditions. The treatment in Chapter 8 is broader in scope, and considers arbitrary MITL formulas.

A parallel body of work proposes the incorporation of probabilities to a TA [95] to yield a probabilistic timed automaton (PTA). Two-player stochastic games are used as an abstraction of the PTA to present tight bounds on the aforementioned probabilities in [96]. Stochastic timed games, defined in [97], assumed two players choosing their actions deterministically, and the environment as a 'half-player' whose actions are probabilistic. The existence of a strategy for one player such that the probability of reaching a set of states under any strategy of the other player and the randomness in the environment is shown to be undecidable in general. The authors of [98] study a two-player stochastic game and showed that it was not possible for a player to have an optimal strategy that guaranteed the 'equilibrium value' against every strategy of the opponent. However, they show the existence of an almost-sure winning strategy for one player against any strategy of the other player. This was not a Markovian policy, since it depends on not only the most recent state, but also on previous states. In all the papers mentioned here, the games are turn-based, and there exists no temporal logic formula that had to be satisfied.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Background

This chapter introduces necessary background on Markov decision processes (MDPs) along with their solution algorithms and stochastic games (SGs). We also present a class of games named Stackelberg games. We additionally introduce background on linear temporal logic (LTL) and metric interval temporal logic (MITL).

3.1 Markov Decision Processes

Markov decision processes (MDPs) are widely used to model the decision making of a single agent in a stochastic environment. At each state of an MDP, the agent (decision maker) takes an action, and the state of the agent changes in a stochastic manner in response to the action. An MDP is formally defined as follows.

Definition 3.1 (Markov Decision Process (MDP))**.** *A Markov Decision Process $\mathcal{M}$ is a tuple $\mathcal{M} = (S, U, P, S_0)$, where*

- *S is a finite set of states.*
- *U is a finite set of actions that can be taken by the agent.*
- *$P : S \times U \times S \to [0, 1]$ is a transition function where $P(s'|s, u)$ is the probability of transitioning from state $s \in S$ to state $s' \in S$ when taking action u.*
- *$S_0 \subseteq S$ is the set of initial states.*

Note that Definition 3.1 focuses on the class of finite MDPs since the state and action spaces S and A are finite. There are other variants of MDPs with infinite state and/or action spaces [99, 100]. We define the set of admissible actions for any state $s \in S$ as $U(s) \subseteq U$, i.e., the set of actions that can be taken by the agent at state s.

Fig. 3-1 gives an example of MDP, modeling a robot navigating in a 2×3 grid world. In this example, the set of states $S = \{1, 2, 3, 4, 5, 6\}$, representing the set of grids in the environment. The robot has four actions $U = \{L, R, U, D\}$, representing 'moving left', 'moving

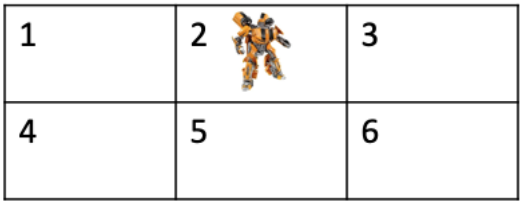


Figure 3-1: An example of MDP modeling a robot navigating in a 2×3 grid world. Each grid is a state of the MDP. The robot has four actions, denoted as $U = \{L, R, U, D\}$, representing 'moving left', 'moving right', 'moving up', and 'moving down'.

right', 'moving up', and 'moving down'. The robot suffers from transition uncertainty when executing the action at each state. For instance, the robot can transition from state 2 to 3 with probability 0.8 when executing action R, and it has probability 0.2 to transition from state 2 to 5. Such transition uncertainty is captured by the transition matrix P.

Given an MDP, there are two categories of strategies (or policies) that can be taken by the agent:

- *Pure strategy*: A pure strategy $\mu : S \to U$ gives the action of the agent as a deterministic function of the state.
- *Mixed strategy*: A mixed strategy $\mu : S \times U \to [0, 1]$, which maps each state and admissible action to a probability distribution over the set of actions $U(s)$ available at state s.

Given a policy μ for an MDP $\mathcal{M}$, MDP reduces to a policy-induced Markov chain (MC), which is defined as follows.

Definition 3.2 (Policy-Induced Markov Chain)**.** *A policy-induced Markov chain from an MDP $\mathcal{M} = (S, U, P, S_0)$ using policy μ is a tuple $\mathcal{M}^\mu = (S, P^\mu, S_0)$, where S and S_0 are as defined in Definition 3.1, and P^μ is the transition probability given as*

$$P^\mu(s'|s) = \sum_{u \in U} P(s'|s, u)\mu(u|s).$$

We say MDP $\mathcal{M}$ is a unichain MDP if for any control policy μ, the policy-induced MC is irreducible, i.e., the probability of reaching any state from any state on the MC is positive.

3.2 Classical Problems and Algorithms on Markov Decision Processes

In this section, we introduce three classical problems on MDPs and their solution algorithms. Given a policy μ for an MDP $\mathcal{M}$, it is evaluated using a reward function $R : S \times U \to \mathbb{R}$.

Reward function R defines the reward the agent can collect via a one-step transition. There are various metrics that have been used to evaluate a policy.

3.2.1 Discounted Total Reward Maximization Problem

The discounted total reward J captures the expected total reward the agent can collect over an infinite time horizon, where the agent gives higher weight that can be obtained in the near future. The discounted total reward is defined as

$$J = \lim_{T\to\infty} \sum_{t=0}^{T-1} \mathbb{E}_{\mu,S_0} \left[\gamma R(s_t, u_t)\right], \tag{3.1}$$

where $\gamma \in (0, 1]$ is a discount factor, $\mathbb{E}_{\mu,S_0}[\cdot]$ represents the expectation with respect to policy μ and the distribution over the initial states S_0, s_t is the state the at t-th step, and u_t is the action taken by the agent following policy μ at t-th step. The discounted total reward maximization problem is then stated as follows.

Problem 3.1 (Discounted Total Reward Maximization)**.** *Given an MDP $\mathcal{M} = (S, U, P, S_0)$ and a reward function $R : S \times U \to \mathbb{R}$, compute a policy μ such that*

$$\max_{\mu} \lim_{T\to\infty} \sum_{t=0}^{T-1} \mathbb{E}_{\mu,S_0} \left[\gamma R(s_t, u_t)\right] \tag{3.2}$$

In the following, we introduce the solution algorithms to Problem 3.1. Let $V^\mu : S \to \mathbb{R}$ be a value function. Here $V^\mu(s)$ gives the expected reward starting from state s following policy μ. It has been shown that V^μ adopts the following recursive property:

$$V^\mu(s) = \sum_{s'} P^\mu(s'|s)(R(s, u) + \gamma V^\mu(s')), \tag{3.3}$$

where P^μ is the transition of the policy-induced Markov chain of $\mathcal{M}$ when using policy μ. Let V^* be the optimal value function. It has been proved that the optimal value function V^* obeys the following *Bellman's optimality condition*:

$$V^*(s) = \max_{\mu} \left[\sum_{s'} P^\mu(s'|s)(R(s, u) + \gamma V^*(s'))\right] \tag{3.4}$$

Motivated by Eqn. (3.4), one can compute V^* via a value iteration algorithm, which is detailed in Algorithm 1. Algorithm 1 iteratively updates value function V_k at each iteration k using Eqn. (3.4) until the value functions between two consecutive iterations are ϵ-close. Algorithm 1 is characterized as follows.

Lemma 3.1 ([101])**.** *Algorithm 1 converges to the unique optimal value V^*.*

Algorithm 1 Value Iteration.

Initialization: $V_0(s) \leftarrow 0$ for all $s \in S$, $k \leftarrow 0$
repeat$V_{k+1}(s) \leftarrow \max_\mu \left[\sum_{s'} P^\mu(s'|s)(R(s,u) + \gamma V_k(s'))\right]$ for all $s \in S$
    $k \leftarrow k + 1$
until $V_{k+1}(s) - V_k(s) \leq \epsilon$ for all $s \in S$
return V_{k+1}

Algorithm 2 Policy Iteration.

Initialization: A policy μ_0, $k \leftarrow 0$
repeat
    Policy Evaluation: Compute $V_k(s)$ such that $V_k(s) = \sum_{s'} P^{\mu_k}(s'|s)(R(s,u) + \gamma V_k(s'))$ for all $s \in S$
    Policy Improvement: Calculate policy μ_{k+1} as $\mu_{k+1} \leftarrow \arg\max_\mu \left[\sum_{s'} P^\mu(s'|s)(R(s,u) + \gamma V_k(s'))\right]$ for all $s \in S$
    $k \leftarrow k + 1$
until $V_{k+1}(s) - V_k(s) \leq \epsilon$ for all $s \in S$
return μ_{k+1}

The optimal value V^* can also be calculated using a method named policy iteration as detailed in Algorithm 2. Similarly, the convergence of Algorithm 2 can be given as follows.

Lemma 3.2 ([101]). *Algorithm 2 converges to the unique optimal value V^*.*

3.2.2 Average Reward per Stage Problem

In this subsection, we consider the average reward per stage problem using a metric called the average reward J_{stg}, which models the reward that the agent can obtain via each one-step transition on average. The average reward is defined as

$$J_{stg} = \lim_{T\to\infty} \frac{1}{T} \mathbb{E}_{\mu,S_0} \left[\sum_{t=0}^{T-1} R(s_t, u_t)\right]. \tag{3.5}$$

The average reward per stage problem is stated as follows.

Problem 3.2 (Average Reward per Stage). *Given an MDP $\mathcal{M} = (S, U, P, S_0)$ and a reward function $R : S \times U \to \mathbb{R}$, compute a policy μ such that*

$$\max_\mu \lim_{T\to\infty} \frac{1}{T} \mathbb{E}_{\mu,S_0} \left[\sum_{t=0}^{T-1} R(s_t, u_t)\right]. \tag{3.6}$$

The optimality condition of Problem 3.2 is given as follows.

Lemma 3.3 ([101]). *Consider a unichain MDP. We have that:*

- *The optimal average reward per stage J^*_{stg} is independent of initial state s_0.*
- *There exists an $|S|$-dimensional vector h^* such that*

$$J^*_{stg} + h^*(s) = \max_{\mu} \Big\{ \sum_{s'} P^{\mu}(s'|s) R(s,u) + \sum_{s' \in S} P^{\mu}(s'|s) h^*(s') \Big\}. \tag{3.7}$$

Eqn. (3.7) is the *Bellman's equation* for average reward per stage problem. The counterparts of value iteration and policy iteration algorithms proposed for discounted total reward maximization problem can be developed for average reward per stage problem using Eqn. (3.7).

3.2.3 Average Reward per Cycle Problem

Average reward per stage problem studies the reward an agent can earn at each stage. In some applications, one may concern the average reward that an agent can earn spanning multiple stages, e.g., a function cycle consisting of multiple stages. This motivates the average reward per cycle problem. Here a cycle is completed if a subset of states $\tilde{S} \subseteq S$ is visited by the agent. Denote the number of cycles that have been completed until stage N as $C(N)$. The average reward per cycle problem is described as follows:

Problem 3.3 (Average Reward per Cycle). *Given an MDP $\mathcal{M} = (S, U, P, S_0)$, a reward function R, and a subset of states $\tilde{S} \subseteq S$ capturing the completions of cycles, compute a policy μ that maximizes*

$$J_{cyc} = \limsup_{N \to \infty} \mathbb{E}_{\mu, S_0} \left\{ \frac{\sum_{t=0}^{N} \sum_{s_{t+1}} P^{\mu}(s_{t+1}|s_t) R(s_t, u_t)}{C(N)} \right\}. \tag{3.8}$$

The optimality condition of average reward per cycle problem is given as follows:

Lemma 3.4 ([26]). *Consider a unichain MDP. We have that:*

- *The optimal average reward per cycle J^*_{cyc} is independent of initial state s_0.*
- *There exists an $|S|$-dimensional vector h^* such that*

$$J^*_{cyc} + h^*(s) = \max_{\mu} \Big\{ \sum_{s'} P^{\mu}(s'|s) R(s,u) + \sum_{s' \in S} P^{\mu}(s'|s) h^*(s') + J^*_{cyc} \sum_{s' \in S} P^{\mu}(s'|s) \Big\}. \tag{3.9}$$

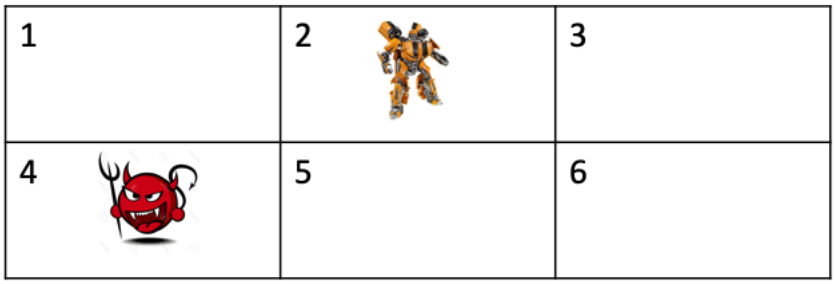


Figure 3-2: An example of MDP modeling a robot navigating in a 2×3 grid world. Each grid is a state of the MDP. The robot and the adversary each has four actions, denoted as $U_1 = U_2 = \{L, R, U, D\}$, representing 'moving left', 'moving right', 'moving up', and 'moving down'.

3.3 Stochastic Games

This section introduces stochastic games (SGs). Different with MDPs which model the decision making of a single agent, SGs concern the decision makings and interaction of two decision makers. An SG is formally defined as follows:

Definition 3.3 (Stochastic Game (SG))**.** *A stochastic game (SG) $\mathcal{G}$ is a tuple $\mathcal{G} = (S, U_1, U_2, P, S_0)$, where*

- *S is a finite set of states.*
- *U_1 is a finite set of actions of agent 1.*
- *U_2 is a finite set of actions of agent 2.*
- *$P : S \times U_1 \times U_2 \times S \to [0, 1]$ is a transition function where $Pr(s'|s, u_1, u_2)$ is the probability of transitioning from state s to state s' when the agents 1 and 2 take actions u_1 and u_2, respectively.*
- *$S_0 \subseteq S$ is the set of initial states.*

For any state $s \in S$, the set of admissible actions to agent 1 and 2 are denoted as $U_1(s)$ and $U_2(s)$, respectively. In an SG, both agents can take either pure strategies or mixed strategies. Different from an MDP, the transition probabilities in an SG is jointly determined by the actions taken by both decision makers.

Fig. 3-2 shows an example of SG, where a robot interacts with an adversary in a 2×3 grid world. The adversary navigates in the grid world, attempting to evade from the robot's pursuit. Both the robot and adversary can take actions in $\{L, R, U, D\}$ to move to the adjacent grids. In this case, a state $s \in S$ of the SG captures the joint location of the robot and adversary, and thus there are 36 states in S. The transition probability matrix P

captures how joint locations of the robot and adversary can change when they take actions in $\{L, R, U, D\}$.

In an SG, the interaction between agents 1 and 2 needs to be understood. Game theory is widely used to model the interaction between multiple agents (also known as players in the context of games). In the sequel, we focus on a class of games named *Stackelberg games*, which have been adopted in a variety of security applications [52, 102].

In a Stackelberg game, agent 1 (also called the leader) commits to a strategy first. Then agent 2 (also known as the follower) observes the strategy of the leader and plays its best response. The information structure under Stackelberg setting can be classified into the following two categories:

- *Turn-based games*: Exactly one player is allowed to take action at each time step. Turn-based games are used to model the case where the players interact in an asynchronous way.

- *Concurrent games*: All the players take actions simultaneously at each time step. Concurrent games are used to model the case where the players interact in an synchronous way.

The policies of the leader and follower in a Stackelberg game are evaluated via the utilities achieved under these policies. Suppose that the leader and follower commit to policies μ and τ, respectively. We denote the utility that the leader gains in a stochastic game $\mathcal{G}$ under leader follower strategy pair (μ, τ)as $\mathcal{Q}_L(\mu, \tau)$. Analogously, the utility of the follower is denoted as $\mathcal{Q}_F(\mu, \tau)$. The solution concept of the Stackelberg games is called the Stackelberg equilibrium (SE), whose definition is given as follows.

Definition 3.4 (Stackelberg Equilibrium (SE))**.** *A pair of strategies (μ, τ) is a Stackelberg equilibrium if leader's strategy μ is optimal given that the follower observes its strategy and plays its best response, i.e.,*

$$\mu = \arg\max_{\mu' \in \boldsymbol{\mu}} \mathcal{Q}_L(\mu', \mathcal{BR}(\mu')),$$

where $\boldsymbol{\mu}$ is the set of all policies of the leader and

$$\mathcal{BR}(\mu') = \{\tau : \tau \in \arg\max \mathcal{Q}_F(\mu', \tau)\}$$

is the best response to leader's strategy μ' played by the follower.

3.4 Formal Specifications: Temporal Logics and Automata

In this section, we introduce background on temporal logics and automata. Temporal logics are concrete and rigorous tools to specify properties for cyber-physical systems. We mainly

focus on two classes of temporal logics, namely linear temporal logic (LTL) and metric interval temporal logic (MITL).

An LTL formula consists of [18]

- a set of atomic propositions Π;
- Boolean operators: negation ($\neg$), conjunction ($\wedge$) and disjunction ($\vee$).;
- temporal operators: next ($\bigcirc$) and until (U).

An LTL formula is defined inductively as

$$\varphi = True \mid \pi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \text{ U } \varphi_2.$$

Other Boolean operators can be defined leveraging connectives $\wedge$ and $\neg$. For example, implication operator $\implies$ is defined as

$$\varphi_1 \implies \varphi_2 \coloneqq \neg\varphi_1 \wedge \varphi_2.$$

Until operator can be used to derive other temporal operators as

$$\Diamond\varphi \coloneqq True\text{U}\varphi, \quad \Box\varphi \coloneqq \neg\Diamond\neg\varphi,$$

where $\Diamond$ and $\Box$ are 'eventually' and 'always' operators.

The semantics of LTL formulas are defined over infinite words in 2^{Π} [18]. Informally speaking, φ is true if and only if φ is true at the current time step. $\psi\text{U}\varphi$ is true if and only if $\psi \wedge \neg\varphi$ is true until φ becomes true at some future time step. $\Box\varphi$ is true if and only if φ is true for the current time step and all the future time. $\Diamond\varphi$ is true if φ is true at some future time. $\bigcirc\varphi$ is true if and only if φ is true in the next time step. A word η satisfying an LTL formula φ is denoted as $\eta \models \varphi$.

Given an LTL formula φ, a deterministic Rabin automaton (DRA) can be constructed to equivalently represent the formula. A DRA is defined as follows.

Definition 3.5 (Deterministic Rabin Automaton)**.** *A deterministic Rabin automaton (DRA) is a tuple $\mathcal{R} = (Q, \Sigma, \delta, q_0, Acc)$, where*

- *Q is a finite set of states.*
- *Σ is a finite set of symbols called alphabet.*
- *$\delta : Q \times \Sigma \to Q$ is the transition function.*
- *$q_0 \in Q$ is the initial state.*
- *$Acc = \{(L(1), K(1)), (L(2), K(2)), \cdots, (L(Z), K(Z))\}$ is a finite set of Rabin pairs such that $L(z), K(z) \subseteq Q$ for all $z = 1, 2, \cdots, Z$ with Z being a positive integer.*

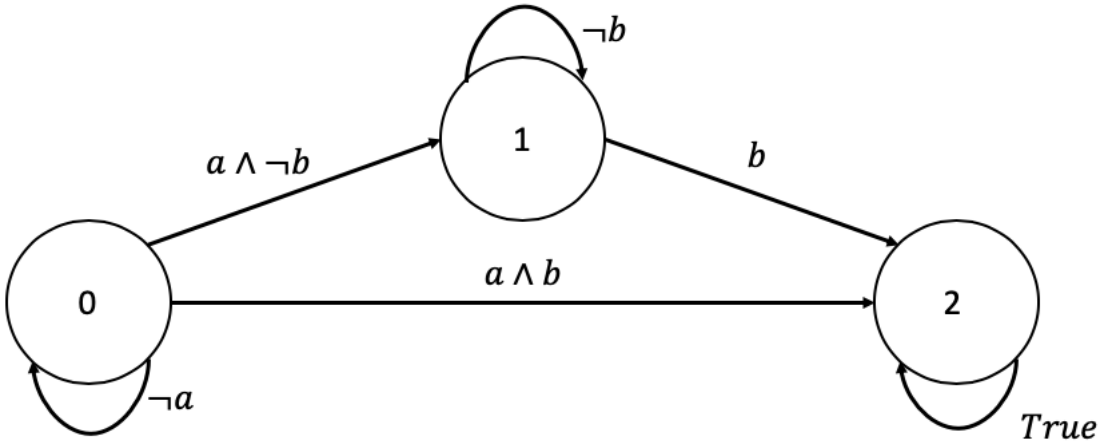


Figure 3-3: The DRA representing LTL formula $\varphi = \diamond(a \wedge \diamond b)$. The states are represented by circles and the transitions are shown by arrows.

Fig. 3-3 presents the DRA of LTL formula $\varphi = \diamond(a \wedge \diamond b)$. LTL specification φ requires to reach a and b sequentially. The state set of the DRA is $Q = \{0, 1, 2\}$, with each being represented as circle in Fig. 3-3. The initial state is $q_0 = 0$. The transitions are represented by arrows. For example, we have that $\delta(0, \neg a) = 0$. There exists one Rabin pair in this example, i.e., $Acc = \{\emptyset, \{2\}\}$.

A fragment of LTL, named co-safe LTL (scLTL) has also been extensively studied. Compared to LTL, scLTL can be interpreted over finite words [18]. Specifically, scLTL is defined as

Definition 3.6 (scLTL)**.** *Any LTL formula that contains only eventually and U temporal operators when written in positive normal form (i.e., negation appears only in front of atomic propositions) is syntactically co-safe.*

Any scLTL formula can be expressd using a deterministic finite automaton (DFA), defined as follows.

Definition 3.7 (Deterministic finite automaton)**.** *A DFA $\mathcal{A}$ is a tuple $\mathcal{A} = (Q, q_0, \Sigma, \delta, F)$, where*

- *Q is a finite set of states.*
- *$q_0 \in Q$ is the initial state.*
- *Σ is alphabet.*
- *$\delta : Q \times \Sigma \to Q$ is the set of transitions.*
- *$F \subseteq Q$ is the set of accepting states.*

A run ρ of a DRA or DFA over a finite input word $\eta = \eta_0\eta_1 \cdots \eta_n$ is a sequence of states $Q^* = q_0q_1 \cdots q_n$ such that $(q_{k-1}, \eta_k, q_k) \in \delta$ for all $0 \leq k \leq n$. A run ρ is accepted by a DRA $\mathcal{R}$ if and only if there exists a pair $(L(z), K(z))$ such that ρ intersects with $L(z)$ finitely many times and intersects with $K(z)$ infinitely often. A run ρ is accepted by a DFA $\mathcal{A}$ if and only if there exists a $q_k \in F$ for some $k \leq n$. Denote the satisfaction of a formula φ by a run ρ as $\rho \models \varphi$.

LTL specifies the order of the occurrence of events. However, it may not be suitable to express time-critical tasks involving deadlines or time intervals. To this end, we consider a class of temporal logics named metric interval temporal logic (MITL), which involves time intervals associated with each temporal operator. An MITL formula is developed from the same set of atomic propositions Π as in LTL and a *time-constrained until* operator U_I, and can be inductively written as:

$$\varphi := True|\pi|\neg\varphi|\varphi_1 \wedge \varphi_2|\varphi_1 \mathrm{U}_I \varphi_2,$$

where $I = [a, b]$ with $a < b$ is a finite time interval.

MITL formulas are interpreted using *timed words*. A timed word is a sequence $\eta = \{(\pi_i, t_i)\}_{i=0}^{\infty}$, where $\pi_i \in 2^{\Pi}$, $t_i \in \mathbb{R}_{\geq 0}$. A time sequence $\{t_i\}_{i=0}^{\infty}$ associated with any timed word η must satisfy the following:

- Monotonicity: for all $i \geq 0$, $t_{i+1} > t_i$;
- Progress: for all $t \in \mathbb{R}_{\geq 0}$, there exists some $t_i \geq t$.

The semantics of MITL are given as follows:

Definition 3.8 (MITL Semantics)**.** *The satisfaction of an MITL formula φ at time t by a timed word η, written $(\eta, t) \models \varphi$, can be recursively defined in the following way:*

- $(\eta, t) \models True$ *if and only if (iff)* $(\eta, 0)$ *is true;*
- $(\eta, t) \models \pi$ *iff* (η, t) *satisfies* π *at time* t*;*
- $(\eta, t) \models \neg\varphi$ *iff* $(\eta, t) \not\models \varphi$*;*
- $(\eta, t) \models \varphi_1 \wedge \varphi_2$ *iff* $(\eta, t) \models \varphi_1$ *and* $(\eta, t) \models \varphi_2$*;*
- $(\eta, t) \models \varphi_1 U_I \varphi_2$ *iff* $\exists k \in I$ *such that* $(\eta, t+k) \models \varphi_2$ *and for all* $m < k, (\eta, t+m) \models \varphi_1$*.*

The satisfaction of the time-constrained operators relies on a set of clock constraints $\Phi(C)$ defined over a clock set C. A clock constraint is inductively defined as

$$\phi = True|False|c \bowtie t|\phi_1 \wedge \phi_2, \tag{3.10}$$

where $\bowtie \in \{\leq, \geq, <, >\}$, $c \in C$ is a clock, and $t \in \mathbb{Q}$ is a non-negative constant. Any MITL formula can be expressed by a timed Büchi automaton (TBA) using the set of clock constraints. A TBA is defined as follows.

Definition 3.9 (Timed Büchi Automaton)**.** *A timed Büchi automaton is a tuple* $\mathcal{T} = (Q, 2^{\Pi}, q_0, C, \Phi(C), E, F)$*, where*

- Q *is a finite set of states.*
- 2^{Π} *is an alphabet over atomic propositions in* Π*.*
- $q_0 \in Q$ *is the initial state.*
- $E \subseteq Q \times Q \times 2^{\Pi} \times 2^{C} \times \Phi(C)$ *is the set of transitions.*
- $F \subseteq Q$ *is the set of accepting states.*

A transition $\langle q, q', a, C', \phi \rangle \in E$ if $\mathcal{A}$ enables the transition from q to q' when a subset of atomic propositions $a \in 2^{\Pi}$ and clock constraints $\phi \in \Phi(C)$ evaluate to true. The clocks in $C' \subseteq C$ are reset to zero after the transition.

Given the set of clocks C, we define the *valuation* of C as $\mathbf{v} : C \to \mathbb{R}^{|C|}$. Given a constant $t \in \mathbb{Q}$, we let $\mathbf{v} + t \coloneqq [\mathbf{v}(1) + t, \cdots, \mathbf{v}(|C|) + t]^{\top}$. The *configuration* of $\mathcal{T}$ is a pair $(q, \mathbf{v})$, with $q \in Q$ and $\mathbf{v}$ being the valuation. A transition $< q, q', a, C', \phi >$ taken after t time units from $(q, \mathbf{v})$ to a configuration $(q', \mathbf{v} + t)$ is written $(q, \mathbf{v}) \xrightarrow{a,t} (q', \mathbf{v}')$, where $\mathbf{v} + t \models \phi$ and $\mathbf{v}'(c) = \mathbf{v}(c) + t$ for all $c \notin C'$. Given an input sequence $a_0, a_1, \cdots$ with $a_i \in \Pi$, we can construct a corresponding sequence of configurations $\rho = (q_0, \mathbf{v}_0) \xrightarrow{a_0,t_0} (q_1, \mathbf{v}_1) \cdots$, called a *run* of $\mathcal{A}$. The run ρ is *feasible* if for all $i \geq 0$ there exists a transition $< q_i, q_{i+1}, a, C_i, \phi >$ in $\mathcal{A}$ such that (i) $\mathbf{v}_0 = \mathbf{0}$, (ii) $\mathbf{0} + t_0 \models \phi_0$, (iii) $\mathbf{v}_1(c) = \mathbf{v}_0(c) + t_0$ for all $c \notin C_0$, and (iv) $\mathbf{v}_i + t_i \models \phi_i$ and $\mathbf{v}_{i+1}(c) = \mathbf{v}_i(c) + t_i$ for all $c \notin C_i$.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

Optimal Secure Control under LTL Constraints

4.1 Introduction

Cyber-physical systems (CPS) are expected to perform increasingly complex tasks in applications including autonomous vehicles, teleoperated surgery, and advanced manufacturing. An emerging approach to designing such systems is to specify a desired behavior using temporal logic formulas, and then automatically synthesize a controller satisfying the given requirements [26, 27, 28, 103, 36, 29].

The integration between cyber and physical components in CPS makes them vulnerable to malicious attacks. An intelligent adversary targeting at CPS can degrade or even invalidate the existing controller synthesis approaches for CPS under temporal logic constraints. Currently, however, automatic synthesis of control systems in adversarial scenarios subject to temporal logic constraints has received limited research attention.

In this chapter, we investigate the problem of computing a control strategy for a probabilistic autonomous system in the presence of an adversary such that the probability of satisfying an LTL specification is maximized. We abstract the system as a stochastic game (SG), which is a generalization of Markov decision process (MDP), and consider a concurrent Stackelberg information structure, in which the adversary and controller take actions simultaneously. We make the following specific contributions:

- We formulate an SG to model the interaction between the CPS and adversary. The SG describes the system dynamics and the effects of the joint input determined by the controller and adversary. We propose a heuristic algorithm to compute the SG given the system dynamics.

- We investigate how to synthesize a control policy that maximizes the worst-case probability of satisfying an arbitrary specification modeled using LTL. We prove that this

problem is equivalent to a zero-sum stochastic Stackelberg game, in which the controller chooses a policy to maximize the probability of reaching a desired set of states and the adversary chooses a policy to minimize that probability.

- We give an algorithm to compute the set of states that the system desires to reach. We then propose an iterative algorithm for constructing an optimal stationary policy. We prove that our approach converges to a Stackelberg equilibrium and characterize the convergence rate of the algorithm.
- We evaluate the proposed approach using a numerical case study. We consider a remotely controlled UAV under deception attack given different LTL specifications. We compare the performance of our proposed approaches with the performance obtained using existing approaches without considering the adversary's presence. The results show that our proposed approach outperforms existing methods.

The remainder of this chapter is organized as follows. Section 4.2 introduces the system model and presents the problem formulation on maximizing the probability of satisfying a given LTL specification. Section 4.3 presents our proposed solution approach. Section 4.4 evaluates our proposed approach and Section 4.5 concludes this chapter.

4.2 Sysem Model and Problem Statement

We consider the following discrete-time system

$$x(t+1) = f(x(t), u_C(t), u_A(t), \vartheta(t)), \ \forall t = 0, 1, \cdots, \tag{4.1}$$

where $x(t)$ is the system state, $u_C(t)$ is the control input from the controller, $u_A(t)$ is the attack signal from the adversary, and $\vartheta(t)$ is stochastic disturbance.

In system (4.1), there exists a strategic adversary that can tamper with the system transition. For instance, an adversary that launches false data injection attack modifies the control input as $u(t) = u_C(t) + u_A(t)$; an adversary that launches denial-of-service attack manipulates the control input as $u(t) = u_C(t) \cdot u_A(t)$, where $u_A(t) \in \{0, 1\}$. Due to the impact from the adversary, the system state transition is jointly determined by the controller and adversary.

We propose a heuristic simulation based algorithm as shown in Algorithm 3, which is generalized from the approaches proposed in [104, 30], to abstract system (4.1) as a stochastic game. The difference between Algorithm 3 and the algorithms in [104, 30] is that Algorithm 3 considers the presence of adversary. Algorithm 3 takes the dynamical system (4.1), the set of sub-regions of state space $\{X_1, \cdots, X_n\}$ and actions as inputs. We observe that the choice of subregions $X_1, \ldots, X_n$ may affect the accuracy of the model, however, choice of the subregions is beyond the scope of this chapter.

Algorithm 3 Algorithm for constructing a stochastic game of a system.

Input: Dynamics (4.1), set of subsets $X_1, \ldots, X_n$
Output: Stochastic game $\mathcal{G} = (S, U_C, U_A, P, S_0, \Pi, \mathcal{L})$
Initialize $\bar{k}$
$S = \{X_1, \ldots, X_n\}$ and $\mathcal{L}$ is determined accordingly
Generate control primitive sets $U_C = \{u_{C_1}, u_{C_2} \cdots, u_{C_\Xi}\}$ and $U_A = \{u_{A_1}, u_{A_2} \cdots, u_{A_\Gamma}\}$
for $i = 1, \ldots, n$ **do**
    for all $u_C \in U_C$ and $u_A \in U_A$ **do**
        for $k = 1, \ldots, \bar{k}$ **do**
            $x \leftarrow$ sampled state in X_i
            $\hat{u}_C, \hat{u}_A \leftarrow$ sampled inputs from u_C, u_A
            $j \leftarrow$ region containing $f(x, \hat{u}_C, \hat{u}_A, \vartheta)$
            Invoke particle filter to approximate transition probabilities P between sub-region i and j for all i and j.
        end for
    end for
end for

We consider the SG $\mathcal{G} = (S, U_C, U_A, P, \Pi, \mathcal{L})$ returned by Algorithm 3 as an abstraction of system (4.1), where S, U_C, U_A, and P are the state set, action set of the controller (agent 1 in Definition 3.3), action set of the adversary (agent 2 in Definition 3.3), and transition probability, respectively. Here Π is a finite set of atomic propositions and $\mathcal{L} : S \to 2^\Pi$ is a labeling function mapping each state to a subset of propositions in Π.

We consider the concurrent Stackelberg setting between the controller and adversary, where the controller plays as the leader and the adversary plays as the follower. The controller first commits to its control strategy μ. The adversary can stay outside for indefinitely long time to observe the strategy of the controller and then chooses its best response τ to the controller's strategy. However, at each time step, both players must take actions simultaneously.

The system is given an LTL specification φ. The objective of the controller is to compute a control strategy μ on SG $\mathcal{G}$ so that the probability of satisfying specification φ is maximized. In contrast, the objective of the adversary is to compute a strategy τ to deviate the system from satisfying the specifications in φ, given the controller's strategy μ. The problem investigated in this chapter is formally stated as follows:

Problem 4.1. *Given a stochastic game $\mathcal{G}$ and an LTL specification φ, compute a control policy μ that maximizes the probability of satisfying specification φ under any adversary policy τ, i.e.,*

$$\max_{\mu} \min_{\tau} \mathbb{P}^{\mu\tau}(\varphi), \tag{4.2}$$

where $\mathbb{P}^{\mu\tau}(\cdot)$ denotes the probability of an event.

We denote the probability of satisfying LTL specification φ as *satisfaction probability*. When the initial state is given, we use $\mathbb{P}^{\mu\tau}(\varphi|s)$ to denote the satisfaction probability when starting from state $s \in S$. Problem 4.1 can be found useful in various applications. We present two applications in the remainder of this section.

Infrastructure Protection in Power System. Problem 4.1 can be used to study attack-defense problems [105] on power systems. The players involved are the power system administrator (controller in Problem 4.1) and adversary. The adversary aims to disrupt the transmission lines in power network, while the administrator deploys resources to protect critical infrastructures or repair damaged infrastructures. Depending on the focus of the administrator, the state may contain bus voltages, bus power injections, network frequency and so on. The actions of the administrator U_C and adversary U_A, respectively, are the actions to protect (by deploying protection or repair resources) and damage (by opening the breakers at ends of) the transmission lines. If an attack is successful, then the transmission line is out of service, which will result in dramatic change on state. Thus the states evolve following the joint actions of the administrator and adversary. Moreover, the probability of the occurrence of events, i.e., the transmission line is out of service, is jointly determined by the actions of adversary and administrator. The specifications that can be given to the system might include reachability (e.g., 'eventually satisfy optimal power flow equation': $\diamond OPF$) and reactivity (e.g., 'if voltage exceeds some threshold, request load shedding from demand side': $\square(voltage\_alarm \implies \bigcirc DR)$).

Networked Control System under Attacks. In the following, we present an example on control synthesis for networked control system under deception attacks. The state set S consists of partitions of the state space of the networked control system. There exists an intelligent and strategic adversary that can compromise the control input of the system by launching deception attack [49]. Typical specifications that are assigned to the system include stability and safety (e.g., 'eventually reach stable status while not reaching unsafe state': $\diamond\square \text{stable} \wedge \square\neg \text{unsafe}$).

4.3 Control Synthesis to Maximize Satisfaction Probability

This section presents the solution approach to Problem 4.1. The proposed solution approach consists of three steps. We first calculate the probability of satisfying the LTL specification φ. We then construct a product SG using SG $\mathcal{G}$ and DRA $\mathcal{R}$, and convert the problem of satisfying φ to the problem of reaching a subset of states, named generalized accepting maximal end component (GAMEC), on the product SG. We finally develop an efficient value iteration algorithm to compute a policy that maximizes the probability of satisfying φ.

4.3.1 Computation of the Probability of Satisfying LTL Specification φ

The policies μ and τ that achieve the max-min value of Eqn. (4.2) can be interpreted as a Stackelberg equilibrium (see Definition 3.4) in a zero-sum Stackelberg game between the

controller and adversary, in which the controller first chooses a randomized policy μ, and the adversary observes μ and selects a policy τ to minimize satisfaction probability $\mathbb{P}^{\mu\tau}(\varphi)$. By Von Neumann's theorem [106], the satisfaction probability at equilibrium must exist. We have the following preliminary lemma.

Lemma 4.1. *Let satisfaction probability with initial state s be $v(s) = \max_\mu \min_\tau \mathbb{P}^{\mu\tau}(\varphi|s)$. Then*

$$v(s) = \max_{\mu} \min_{\tau} \sum_{u_C \in U_C} \sum_{u_A \in U_A} \sum_{s' \in S} \mu(s, u_C)\tau(s, u_A)v(s')P(s'|s, u_C, u_A). \tag{4.3}$$

Conversely, if $v(s)$ satisfies Eqn. (4.3), then $v(s) = \max_\mu \min_\tau \mathbb{P}^{\mu\tau}(\varphi|s)$. Moreover, the satisfaction probability v is unique.

Proof. In the following, we will first show the forward direction. We let $n = |S|$ and define three operators $T_{\mu\tau} : [0,1]^n \to [0,1]^n$, $T_\mu : [0,1]^n \to [0,1]^n$, and $T : [0,1]^n \to [0,1]^n$ as

$$\begin{aligned}(T_{\mu\tau}v)(s) &= \sum_{s'} P(s'|s, \mu, \tau)v(s') \\ (T_\mu v)(s) &= \min_\tau \sum_{s'} P(s'|s, \mu, \tau)v(s') \\ (Tv)(s) &= \max_\mu \min_\tau \sum_{s'} P(s'|s, \mu, \tau)v(s')\end{aligned}$$

where $P(s'|s, \mu, \tau) = \sum_{u_C \in U_C} \sum_{u_A \in U_A} \mu(s, u_C)\tau(s, u_A)P(s'|s, u_C, u_A)$ represents the transition probability induced by policies μ and τ. Suppose that μ is a Stackelberg equilibrium with $v(s)$ equal to the satisfaction probability for state s, and yet Eqn. (4.3) does not hold. We have that $v = T_\mu v$, since v is the optimal policy for the MDP defined by the policy μ [101]. On the other hand, $T_\mu v \leq Tv$. Composing T and T_μ by k times and taking the limit as k approaches infinity yields $v = \lim_{k\to\infty} T_\mu^k v \leq \lim_{k\to\infty} T^k v \triangleq v^*$. The convergence of $T^k v$ to a fixed point v^* follows from the fact that T is a bounded and monotone non-decreasing operator. Furthermore, choosing the policy μ at each state as the maximizer of Eqn. (4.3) yields a policy with satisfaction probability v^*. Hence $v \leq v^*$. If $v(s) = v^*(s)$ for all states s, then Eqn. (4.3) is satisfied, contradicting the assumption that the equation does not hold. On the other hand, if $v(s) < v^*(s)$ for some state s, then μ is not a Stackelberg equilibrium.

We next show that vector v satisfying Eqn. (4.3) is unique. Since every Stackelberg equilibrium satisfies Eqn. (4.3), if vector v is unique, then vector v must be a Stackelberg equilibrium. Suppose that uniqueness does not hold, and let μ and μ' be Stackelberg equilibrium policies with corresponding satisfaction probabilities v and v'. We have that $v = Tv \geq T_{\mu'}v$. Composing k times and taking the limit as k tends to infinity, we have $v = \lim_{k\to\infty} T^k v \geq \lim_{k\to\infty} T_{\mu'}^k v = v'$. By the same argument, $v' \geq v$, implying that $v = v'$ and thus uniqueness holds. □

By Lemma 4.1, we have that the satisfaction probability for some state s can be computed as the linear combination of the satisfaction probabilities of its neighbor states, where the coefficients are the transition probabilities jointly determined by policies μ and τ. Lemma 4.1 provides us the potential to apply iterative algorithm to compute the satisfaction probability.

4.3.2 Equivalence to the Problem of Maximizing Reachability Probability

In the following, we first construct a product SG which captures the transition probabilities in SG and the satisfaction of LTL specification φ. A product SG is defined as follows.

Definition 4.1. (Product SG): *Given an SG* $\mathcal{G} = (S, U_C, U_A, P, , \Pi, \mathcal{L})$ *and a DRA* $\mathcal{R} = (Q, \Sigma, \delta, q_0, Acc)$ *representing LTL specification* φ*, a (labeled) product SG is a tuple* $\mathcal{G}_{prod} = (S_{prod}, U_C, U_A, P_{prod}, Acc_{prod})$*, where*

- $S_{prod} = S \times Q$ *is a finite set of states.*
- U_C *is a finite set of control inputs.*
- U_A *is a finite set of attack signals.*
- $P_{prod}((s', q')|(s, q), u_C, u_A) = P(s'|s, u_C, u_A)$ *if* $\delta(q, \mathcal{L}(s')) = q'$.
- $Acc_{prod} = \{(L_{prod}(1), K_{prod}(1)), (L_{prod}(2), K_{prod}(2)), \cdots, (L_{prod}(Z), K_{prod}(Z))\}$ *is a finite set of Rabin pairs such that* $L_{prod}(z), K_{prod}(z) \subseteq S_{prod}$ *for all* $z = 1, 2, \cdots, Z$ *with* Z *being a positive integer. In particular, a state* $(s, q) \in L_{prod}(z)$ *if and only if* $q \in L(z)$*, and a state* $(s, q) \in K_{prod}(z)$ *if and only if* $q \in K(z)$.

Definition 4.1 bridges SG $\mathcal{G}$ and LTL specification φ in the following three aspects. First, since the transition probability is determined by $\mathcal{G}$ and the satisfaction condition is determined by $\mathcal{R}$, the satisfaction probability of φ on $\mathcal{G}$ is equal to the satisfaction probability of φ on the product SG $\mathcal{G}_{prod}$. Second, we can generate the corresponding path $s_0 s_1 \cdots$ on $\mathcal{G}$ given a path $(s_0, q_0)(s_1, q_1) \cdots$ on the product SG $\mathcal{G}_{prod}$. Finally, given a control policy μ synthesized on the product SG $\mathcal{G}_{prod}$, a corresponding control policy μ' on $\mathcal{G}$ is obtained by letting $\mu'(s_i) = \mu((s_i, q))$ for all time step i [18, 30]. Due to these one-to-one correspondence relationships, in the following, we analyze Problem 4.1 on the product SG $\mathcal{G}_{prod}$ and present an algorithm to compute the optimal control policy. When the context is clear, we use s to represent state $(s, q) \in S_{prod}$ in the remainder of this chapter.

We solve the problem of maximizing the satisfaction probability by converting it to the problem of maximizing the probability of reaching a certain subset of states of the product SG. We name this subset of states as Generalized Accepting Maximal End Component (GAMEC), which is generalized from accepting maximal end component (AMEC) on MDP. GAMEC is defined as the set of states when starting from which the satisfaction probability of LTL specification φ is one. We give the definition of GAMEC in the following:

Definition 4.2. (Sub-SG): *A sub-SG of an SG* $\mathcal{G} = (S, U_C, U_A, Pr, s_0, \Pi, \mathcal{L})$ *is a pair of states and actions* (C, D) *where* $\emptyset \neq C \subseteq S$ *is a set of states, and* $D : C \to 2^{U_C(s)}$ *is an enabling function such that* $D(s) \subseteq U_C(s)$ *for all* $s \in C$ *and* $\{s' | P(s'|s, u_C, u_A) > 0, \forall u_A \in U_A(s), s \in C\} \subseteq C$.

By Definition 4.2, we have that a sub-SG is also an SG. Given Definition 4.2, a Generalized Maximal End Component (GMEC) is defined as follows:

Definition 4.3. *A Generalized End Component (GEC) is a sub-SG* (C, D) *such that the underlying digraph* $G_{(C,D)}$ *of sub-SG* (C, D) *is strongly connected. A GMEC is a GEC* (C, D) *such that there exists no other GEC* $(C', D') \neq (C, D)$*, where* $C \subseteq C'$ *and* $D(s) \subseteq D'(s)$ *for all* $s \in C$.

Definition 4.4. *A GAMEC on the product SG* $\mathcal{G}_{prod}$ *is a GMEC if there exists some* $(L_{prod}(z), K_{prod}(z)) \in Acc_{prod}$ *such that* $L_{prod}(z) \cap C = \emptyset$ *and* $K_{prod}(z) \subseteq C$.

By Definition 4.4, we have that a set of states constitutes a GAMEC if there exists a control policy such that for any initial states in the GAMEC, the system remains in the GAMEC with probability one and thus the specification is satisfied with probability one. We denote the set of GAMECs as $\mathcal{C}$, and the set of states that constitute GAMEC as accepting states $\mathcal{E}$. Algorithm 4 is used to compute the set of GAMECs. Given a product SG $\mathcal{G}$, a set of GAMECs $\mathcal{C}$ can be initialized as $C = \emptyset$ and $D(s) = U_C(s)$ for all s. Also, we define a temporary set $\mathcal{C}_{temp}$ which is initialized as $\mathcal{C}_{temp} = S_{prod}$. Then from line 8 to line 17, we compute a set of states R that should be removed from GMEC. The set R is first initialized to be empty. Then for each state s in each non-trivial strongly connected component (SCC) of the underlying diagraph, i.e., the SCC with more than one states, we modify the admissible actions at state s by keeping the actions that can make the system remain in C under any adversary action. If there exists no such admissible action at state s, then the state s is added into R. From line 18 to line 26, we examine if there exists any state s' in current GMEC that will steer the system into states in R. In particular, by taking action u_C at each state s', if there exists some adversary action u_A such that the system is steered into some state $s \in R$, then u_C is removed from $U_C(s')$. Moreover, if there exists no admissible action at state s', then s' is added to R. Then we update the GMEC set as shown from line 27 to line 32. This procedure is repeated until no further update can be made on GMEC set. Line 34 to line 40 is to find the GAMEC following Definition 4.4. Given the set of GAMECs $\mathcal{C} = \{(C_1, D_1), \cdots, (C_h, D_h), \cdots, (C_{|\mathcal{C}|}, D_{|\mathcal{C}|})\}$ returned by Algorithm 4, the set of accepting states $\mathcal{E}$ is computed as $\mathcal{E} = \cup_{h=1}^{|\mathcal{C}|} C_h$.

Provided the definition of GAMEC, we then show that the max-min satisfaction probability is equivalent to maximizing (over μ) the worst-case probability of reaching the set of accepting states $\mathcal{E}$. We name the probability of reaching the set of accepting states $\mathcal{E}$ as *reachability probability*. In the following, we formally prove the equivalence between the worst-case satisfaction probability of Eqn. (4.2) and the worst-case reachability probability.

Algorithm 4 Computing the set of GAMECs $\mathcal{C}$.

procedure COMPUTE_GAMEC($\mathcal{G}_{prod}$)
  Input: Product SG $\mathcal{G}_{prod}$
  Output: Set of GAMECs $\mathcal{C}$
  Initialization: Let $D(s) = U_C(s)$ for all $s \in S_{prod}$. Let $\mathcal{C} = \emptyset$ and $\mathcal{C}_{temp} = \{S_{prod}\}$
  repeat
    $\mathcal{C} = \mathcal{C}_{temp}$, $\mathcal{C}_{temp} = \emptyset$
    for $C \in \mathcal{C}$ **do**
      $R = \emptyset$ ▷ R is the set of states that should be removed
      Let $SCC_1, \cdots, SCC_n$ be the set of nontrivial strongly connected components (SCC) of the underlying diagraph $H_{(C,D)}$
      for $i = 1, \cdots, n$ **do**
        for each state $s \in SCC_i$ **do**
          $D(s) = \{u_C \in U_C(s) | s' \in C$ where $P_{prod}(s'|s, u_C, u_A) > 0, \ \forall u_A \in U_A(s)\}$
          if $D(s) = \emptyset$ **then**
            $R = R \cup \{s\}$
          end if
        end for
      end for
      while $R \neq \emptyset$ **do**
        dequeue $s \in R$ from R and C
        if there exist $s' \in C$ and $u_C \in U_C(s')$ such that $P_{prod}(s|s', u_C, u_A) > 0$ under some $u_A \in U_A(s')$ **then**
          $D(s') = D(s') \setminus \{u_C\}$
          if $D(s') = \emptyset$ **then**
            $R = R \cup \{s'\}$
          end if
        end if
      end while

Algorithm 5 Algorithm 4 Contd.

```
        for i = 1, ··· , n do
            if C ∩ SCC_i ≠ ∅ then
                C = C_temp ∪ {C ∩ SCC_i}
            end if
        end for
    end for
    until C = C_temp
    for  doC ∈ C
        for (L_prod(z), K_prod(z)) ∈ Acc_prod do
            if L_prod(z) ∩ C ≠ ∅ or K_prod(z) ⊈ C then
                C = C \ C
            end if
        end for
    end for
    return C
end procedure
```

Proposition 4.1. *For any stationary control policy μ and initial state s, the minimum probability over all stationary adversary policies of satisfying the LTL formula is equal to the minimum probability over all stationary policies of reaching $\mathcal{E}$, i.e., given any stationary policy μ, we have*

$$\min_{\tau} \mathbb{P}^{\mu\tau}(\varphi|s) = \min_{\tau} \mathbb{P}^{\mu\tau}(\textit{reach } \mathcal{E}|s), \tag{4.4}$$

where $\mathbb{P}^{\mu\tau}(\textit{reach } \mathcal{E}|s)$ is the probability of reaching $\mathcal{E}$ under policies μ and τ.

Proof. By Definition of $\mathcal{E}$, if the system reaches $\mathcal{E}$, then φ is satisfied for a maximizing policy μ. Thus $\min_\tau \mathbb{P}^{\mu\tau}(\text{reach } \mathcal{E}) = \min_\tau \mathbb{P}^{\mu\tau}(\varphi)$.

Suppose that for some control policy μ and initial state s_0,

$$\min_{\tau} \mathbb{P}^{\mu\tau}(\varphi|s_0) > \min_{\tau} \mathbb{P}^{\mu\tau}(\text{reach } \mathcal{E}|s_0), \tag{4.5}$$

and let τ be a minimizing stationary policy for the adversary. The policies μ and τ induce an MC on the state space (See Definition 3.2). By model checking algorithms on MC [18], the probability of satisfying φ from s_0 is equal to the probability of reaching a bottom strongly connected component (BSCC) that satisfies φ. By our assumption there exists a BSCC, denoted SCC_0, that is reachable from s_0, disjoint from $\mathcal{E}$, and yet satisfies $\mathbb{P}^{\mu\tau}(\varphi|s) = 1$ for all $s \in S_0$ (if this were not the case, then Eqn. (4.5) would not hold).

Choose a state $s \in SCC_0$. Since $s \notin \mathcal{E}$, there exists a policy $\hat{\tau}$ such that $\mathbb{P}^{\mu\hat{\tau}}(\varphi|s) < 1$ using policies μ and $\hat{\tau}$. Create a new adversary policy τ_1 as $\tau_1(s') = \hat{\tau}(s')$ for all $s' \in SCC_0$ and $\tau_1(s') = \tau(s')$ otherwise. This policy induces a new MC on the state space. Furthermore, since only the outgoing transitions from SCC_0 are affected, the success probabilities of all

sample paths that do not reach SCC_0 are unchanged.

If there exists any state s' that is reachable from s in the new chain with $\mathbb{P}^{\mu\tau_1}(\varphi|s') < 1$ under policies μ and τ_1, then the policy τ_1 strictly reduces the probability of satisfying φ, thus contradicting the assumption that τ is a minimizing policy. Otherwise, let SCC_1 denote the set of states that are reachable from s under μ and τ_1 and are disjoint from $\mathcal{E}$ (this set must be non-empty; otherwise, the policy $\hat{\tau}$ would lead to $\mathbb{P}^{\mu\hat{\tau}}(\varphi|s) = 1$, a contradiction). Construct a new policy τ_2 by $\tau_2(s') = \hat{\tau}(s')$ if $s' \in S_1$ and $\tau_2(s') = \tau(s')$ otherwise. Proceeding inductively, we derive a sequence of policies τ_k that satisfy $\mathbb{P}^{\mu\tau_k}(\varphi) \leq \mathbb{P}^{\mu\tau}(\varphi)$. This process terminates when either $\mathbb{P}^{\mu\tau_k}(\varphi|s_0) < \mathbb{P}^{\mu\tau}(\varphi|s_0)$, contradicting the minimality of τ, or when $\mathbb{P}^{\mu\tau_k}(\varphi|s'') = \mathbb{P}^{\mu\hat{\tau}}(\varphi|s'')$ for all s'' that are reachable from s under $\hat{\tau}$. The latter case, however, implies that $\mathbb{P}^{\mu\hat{\tau}}(\varphi|s) = 1$, contradicting the definition of $\hat{\tau}$. □

Algorithm 6 Modifying product SG $\mathcal{G}_{prod}$.

1: **Input**: Product SG $\mathcal{G}_{prod}$, the set of GAMECs $\mathcal{C}$
2: **Output:** Modified product SG $\mathcal{G}_{prod}$
3: $S_{prod} := S_{prod} \cup \{dest\}$,$U_C(s) := U_C(s) \cup \{d\},\ \forall s \in S_{prod}$
4: $P_{prod}(dest|s, d, u_A) = 1$ for all $s \in \mathcal{E} \cup \{dest\}$ and $u_A \in U_A(s)$

Proposition 4.1 implies that the problem of maximizing the worst-case success probability can be mapped to a reachability problem on the product SG $\mathcal{G}_{prod}$, where $\mathcal{G}_{prod}$ is modified following Algorithm 6. A dummy state *dest* is added into the state space of S_{prod}. All transitions starting from a state in GAMECs are directed to state *dest* with probability one regardless of the actions taken by the adversary. The transition probabilities and action spaces of all other nodes are unchanged. We observe that the reachability probability remains unchanged after applying Algorithm 6. Hence the satisfaction probability remains unchanged. Moreover, the one-to-one correspondence of control policy still holds for states outside $\mathcal{E}$. Therefore, Problem 4.1 is then equivalent to

$$\max_{\mu} \min_{\tau} \mathbb{P}^{\mu\tau}(\text{reach } dest) \tag{4.6}$$

Then, the solution to Eqn. (4.2) can be obtained from the solution to Eqn. (4.6) by following the optimal policy μ^* for Eqn. (4.6) at all states not in $\mathcal{E}$. The control policy for states in $\mathcal{E}$ can be any probability distribution over the set of enabled actions in each GAMEC.

4.3.3 Value Iteration Algorithm for Control Synthesis

Due to Proposition 4.1, in the following we focus on solving the problem (4.6). Our approach to solving Eqn. (4.6) is to first compute a value vector $v \in \mathbb{R}^{|S_{prod}|}$, where $v(s) = \max_{\mu} \min_{\tau} \mathbb{P}^{\mu\tau}(\text{reach } dest|s)$. By Lemma 4.1, the optimal policy can then be obtained from

Algorithm 7 Algorithm for computing a policy that maximizes the probability of satisfying φ.

Input: product SG $\mathcal{G}_{prod}$, the set of GAMECs $\mathcal{C}$
Output: Vector $v \in \mathbb{R}^{|S_{prod}|}$, where $v(s) = \max \min P^{\mu\tau}_{prod}(\text{reach } dest | s_0 = s)$
Initialization: $v^0 \leftarrow 0$, $v^1(s) \leftarrow 1$ for $s \in \mathcal{E}$, $v^1(s) \leftarrow 0$ otherwise, $k \leftarrow 0$
while $\max\{|v^{k+1}(s) - v^k(s)| : s \in S_{prod}\} > \delta$ **do**
    $k \leftarrow k + 1$
    for $s \notin \mathcal{E}$ **do**
        Compute v as $v^{k+1}(s) \leftarrow \max_\mu \min_\tau \Big\{ \sum_{s'} \sum_{u_C \in U_C(s)} \sum_{u_A \in U_A(s)} v(s') \mu(s, u_C) \tau(s, u_A) P_{prod}(s' | s, u_C, u_A) \Big\}$
    end for
end while
return v

v by choosing the distribution μ that solves the optimization problem (4.3) at each state s. Algorithm 7 gives a value iteration based algorithm for computing v. The idea of the algorithm is to initialize v to be zero except on states in $\mathcal{E}$, and then greedily update $v(s)$ at each iteration by computing the optimal Stackelberg policy at each state. The algorithm terminates when a stationary v is reached.

The following theorem shows that Algorithm 7 guarantees convergence to a Stackelberg equilibrium.

Theorem 4.1. *There exists v^∞ such that for any $\epsilon > 0$, there exists δ and $\bar{k}$ such that $\|v^k - v^\infty\|_\infty < \epsilon$ for $k > \bar{k}$. Furthermore, v^∞ satisfies the conditions of v in Lemma 4.1.*

Proof. We first show that, for each s, the sequence $v^k(s) : k = 1, 2, \ldots$, is bounded and monotone. Boundedness follows from the fact that, at each iteration, $v^k(s)$ is a convex combination of the states of its neighbors, which are bounded above by 1. To show monotonicity, we induct on k. Note that $v^1(s) \geq v^0(s)$ and $v^2(s) \geq v^1(s)$ since $v^1(s) = 0$ for $s \notin \mathcal{E}$ and $v^k(s) \equiv 1$ for $s \in \mathcal{E}$.

Let μ^k denote the control policy at step k. We have

$$v^{k+1}(s) \geq \min_\tau \sum_{u_C \in U_C(s)} \sum_{u_A \in U_A(s)} \sum_{s' \in S} v^k(s') \mu^k(s, u_C) \tau(s, u_A) P_{prod}(s' | s, u_C, u_A) \tag{4.7a}$$

$$\geq \min_\tau \sum_{u_C \in U_C(s)} \sum_{u_A \in U_A(s)} \sum_{s' \in S} v^{k-1}(s') \mu^k(s, u_C) \tau(s, u_A) P_{prod}(s' | s, u_C, u_A) \tag{4.7b}$$

$$= v^k(s) \tag{4.7c}$$

Eqn. (4.7a) follows because the value of $v^{k+1}(s)$, which corresponds to the maximizing policy, dominates the value achieved by the particular policy μ^k. Eqn. (4.7b) holds by induction,

since $v^k(s') \geq v^{k-1}(s')$ for all s'. Finally, Eqn. (4.7c) holds by construction of μ^k for each state s. Hence $v^k(s)$ is monotone in k. We therefore have that $v^k(s)$ is a bounded monotone sequence, and hence converges by the monotone convergence theorem. Let v^∞ denote the vector of limit points, so that we can select δ sufficiently small (to prevent the algorithm from terminating before convergence) and $\bar{k}$ large in order to satisfy $\|v^k - v^\infty\|_\infty < \epsilon$.

We now show that v^∞ is a Stackelberg equilibrium. Since $v^k(s)$ converges, it is a Cauchy sequence and thus for any $\epsilon > 0$, there exists $\bar{k}$ such that $k > \bar{k}$ implies that $|v^k(s) - v^{k+1}(s)| < \epsilon$. By construction, this is equivalent to

$$\left| v^k(s) - \max_{\mu} \min_{\tau} \sum_{u_C \in U_C(s)} \sum_{u_A \in U_A(s)} \sum_{s' \in S} \left[v^{k-1}(s') \mu(s, u_C) \tau(s, u_A) P_{prod}(s' | s, u_C, u_A) \right] \right| < \epsilon,$$

and hence v^∞ is within ϵ of a Stackelberg equilibrium for every $\epsilon > 0$. □

While this approach guarantees asymptotic convergence to a Stackelberg equilibrium, there is no guarantee on the rate of convergence. By modifying line 8 of the algorithm so that $v^{k+1}(s)$ is updated if

$$\max_{\mu} \min_{\tau} \sum_{u_C \in U_C(s)} \sum_{u_A \in U_A(s)} \sum_{s' \in S} \left[v(s') \mu(s, u_C) \tau(s, u_A) P_{prod}(s' | s, u_C, u_A) \right] > (1 + \epsilon) v^k(s) \tag{4.8}$$

and is constant otherwise, we derive the following result on the termination time.

Proposition 4.2. *The ϵ-relaxation of (4.8) converges to a value of v satisfying $\max\{|v^{k+1}(s) - v^k(s)| : s \in S_{prod}\} < \epsilon$ within $n \max_s \left\{ \log\left(\frac{1}{v^0(s)}\right) / \log(1+\epsilon) \right\}$ iterations, where $v^0(s)$ is the smallest positive value of $v^k(s)$ for $k = 0, 1, \ldots$.*

Proof. After N updates, we have that $v^N(s) \geq (1+\epsilon)^N v^0(s)$. Hence for each s, $v(s)$ will be incremented at most $\max_s \left\{ \log\left(\frac{1}{v^0(s)}\right) / \log(1+\epsilon) \right\}$ times. Furthermore, we have that at least one $v(s)$ must be updated at each iteration, thus giving the desired upper bound on the number of iterations. By definition of (4.8), the set that is returned satisfies $|v^{k+1}(s) - v^k(s)| < \epsilon v^k(s) < \epsilon$. □

4.4 Case Study

In this section, we present a case study to demonstrate our proposed method. In this case study, we focus on the application of remotely controlled UAV, which conducts package delivery service by navigating in a discrete bounded grid environment. The UAV carries multiple packages and is required to deliver the packages to pre-given locations in particular order (e.g., the solution of a travelling salesman problem). The label of each state is shown in Fig. 4-1a. The UAV is required to deliver packages to three locations 'dest1', 'dest2',

State	$(7,8)$	$(8,8)$	$(13,8)$	$(14,8)$	$(7,13)$	$(8,13)$	$(13,13)$	$(14,13)$
$\mathbb{P}^{\mu\tau}(\varphi)$	0.6684	0.6028	0.5915	0.4893	0.8981	0.7126	0.6684	0.6028
$\mathbb{P}^{\tilde{\mu}\tilde{\tau}}(\varphi)$	0.3619	0.3182	0.2878	0.1701	0.6146	0.5112	0.3619	0.3182
Improvement	84.69%	89.44%	105.52%	187.65%	46.13%	39.40%	84.69%	89.44%

Table 4.1: Comparison of probabilities of satisfying specification φ when starting from the states located in intersections using proposed approach and approach without considering the adversary.

and 'dest3' in this particular order after departing from its 'home'. Then it has to return to 'home' and stay there forever. Also during this delivery service, the UAV should avoid colliding with obstacle areas marked as black in Fig. 4-1a to Fig. 4-1c. The LTL formula is written as $\varphi = \text{home} \wedge \diamondsuit(\text{dest1} \wedge \diamondsuit(\text{dest2} \wedge \diamondsuit\text{dest3})) \wedge \diamondsuit\square\text{home} \wedge \square\neg\text{obstacle}$.

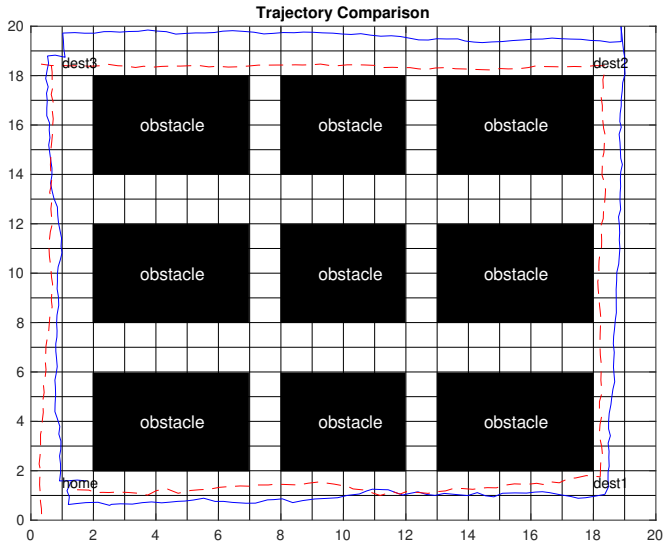
We compare the control policy obtained using the proposed approach with that synthesized without considering the presence of the adversary. In Fig. 4-1a, we present the sample trajectories obtained using these approaches. The solid line shows a sample trajectory obtained by using the proposed approach, and the dashed line shows the trajectory obtained by using the control policy synthesized without considering the presence of adversary. To demonstrate the resilience of the proposed approach, we let the states located in the intersections be labelled as 'home' and hence are set as the initial states. We compare the probability of satisfying the specification φ using the proposed approach and the approach without considering the adversary in Fig. 4-1b and Fig. 4-1c, respectively. We observe that the control policy synthesized using the proposed approach has higher probability of satisfying specification φ. The detailed probability of satisfying specification φ is listed in Table 4.1. Denote the probability of satisfying specification φ using the proposed approach and the approach without considering the adversary as $\mathbb{P}^{\mu\tau}(\varphi)$ and $\mathbb{P}^{\tilde{\mu}\tilde{\tau}}(\varphi)$, respectively. By using the proposed approach, the average of the improvements of the probability of satisfying the given specification starting from intersection states achieves $(\mathbb{P}^{\mu\tau}(\varphi) - \mathbb{P}^{\tilde{\mu}\tilde{\tau}}(\varphi))/\mathbb{P}^{\tilde{\mu}\tilde{\tau}}(\varphi) = 90.87\%$.

Given the SG and DRA associated with specification φ are created within 1 and 0.01 second, respectively. The computation of product SG took 80 seconds. The product SG has 2000 states and 41700 transitions. It took 45 seconds to compute the control policy on a Macbook Pro with 2.6GHz Intel Core i5 CPU and 8GB RAM.

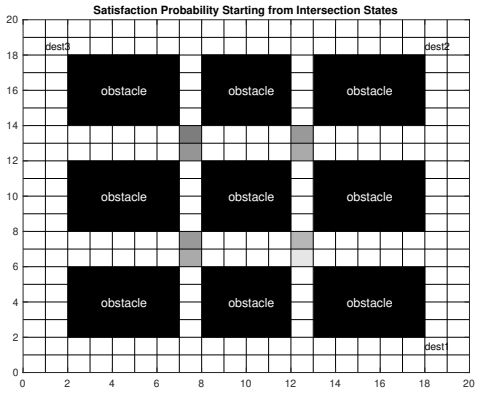
4.5 Conclusion

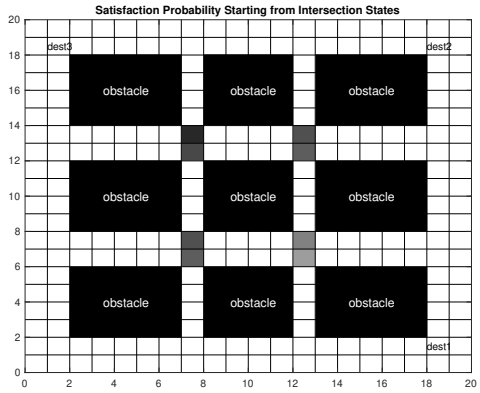
In this chapter, we investigated the problem of computing a control policy to maximize the probability of satisfying a given LTL specification in the presence of a malicious adversary. We assumed that the adversary can initiate malicious attacks on the system by observing the control policy of the controller and choosing an intelligent strategy. A stochastic Stackelberg game was formulated to model the interaction between the controller and adversary. We characterized the satisfaction probability when starting from each state as a linear combina-

tion of those of its neighbor states. We proved that maximizing the satisfaction probability is equivalent to maximizing the probability of reaching a set of states named GAMECs. We developed an efficient value iteration algorithm to compute the equilibrium policy. The potential ways to reduce the computation complexity include exploring the symmetry of the environment, and applying receding horizon based control framework. In future work, we will consider non-stationary control and adversary policies.



(a)





(b) (c)

Figure 4-1: Comparison of the proposed approach and the approach without considering the presence of the adversary. Fig. 4-1a gives the trajectories obtained using two approaches. The solid blue line is the trajectory obtained using the proposed approach, while the dashed red line represents the trajectory obtained using the approach without considering the presence of the adversary. Fig. 4-1b and Fig. 4-1c present the probability of satisfying the LTL specification using the proposed approach and the approach without considering the adversary when the initial state is set as each of the states lands in the intersections of the grid world, respectively. The shade of gray level at the intersection states corresponds to the satisfaction probability, with black being probability 0 and white being probability 1.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 5

Secure Control under LTL Constraints with Minimal Invariant Constraint Violation Rate

5.1 Introduction

Chapter 4 studies the problem of maximizing the probability of satisfying a given LTL specification. As the LTL formula becomes increasingly complex, the system may not be able to satisfy it, and thus the approach proposed in Chapter 4 renders a trivial policy with satisfaction probability zero. To this end, one may be interested in synthesizing a control policy so that (i) allows partial violation of the given LTL specification, and (ii) maximizes the probability of satisfying the rest part of the specification. Moreover, the control policy should minimize the cost incurred due to partially violating the specification.

In this chapter, we focus on a subclass of LTL specification that combines an arbitrary LTL specification with an invariant constraint using logical and connectives, where an invariant constraint requires the system to always satisfy some property. The specification of interest is commonly required for CPS, where the arbitrary LTL specification can be used to model properties such as liveness and the invariant property can be used to model safety property. We relax the specification by allowing violations on the invariant constraint and we select a control policy that minimizes the average rate at which invariant property violations occur while maximizing the probability of satisfying the LTL specification. We make the following specific contributions:

- We formulate the problem of computing a stationary control policy that minimizes the rate at which invariant constraint violations occur under the constraint that an LTL specification must be satisfied with maximum probability. We capture the impact of invariant constraint violation as average cost per cycle, where a cycle completion is equivalent to satisfying the LTL specification.

- We prove that this problem is equivalent to a zero-sum Stackelberg game in which the controller selects a control policy that minimizes the average violation cost and the adversary selects a policy that maximizes such cost. We solve the problem by building up the connections with a generalized average cost per stage problem.
- We propose a novel policy iteration algorithm to compute an optimal stationary control policy. We prove the optimality and convergence of the proposed algorithm.
- We evaluate the proposed approach using a numerical case study on a remotely controlled UAV performing reach-avoid task. We compare the performance of our proposed approaches with the performance obtained using existing approaches without considering the adversary's presence. The results show that our proposed approach outperforms existing methods.

The remainder of this chapter is organized as follows. Section 5.2 introduces the system model and problem formulation. Section 5.3 presents our proposed solution approach. Section 5.4 evaluates our proposed approach and Section 5.5 concludes this chapter.

5.2 Sysem Model and Problem Statement

The system and adversary models considered in this chapter are as in Chapter 4. We consider an SG $\mathcal{G} = (S, U_C, U_A, P, \Pi, \mathcal{L})$ with concurrent Stackelberg setting between the controller and adversary.

When the system is given an LTL specification, it might be impossible for the system to satisfy the specification due to the presence of the adversary, i.e., $\max_\mu \min_\tau \mathbb{P}^{\mu\tau}(\varphi) = 0$, where $\mathbb{P}^{\mu\tau}(\varphi)$ denotes the probability of satisfying φ under policies μ and τ. Thus, we need to relax the specification by allowing partial violations of the specification.

In this chapter, we focus on a subclass of specifications of the form $\varphi = \varphi_1 \wedge \psi$, where φ_1 is an arbitrary LTL formula and ψ is an *invariant constraint*. An invariant constraint requires the system to always satisfy some property. The general LTL formula φ_1 can be used to model any arbitrary properties such as liveness $\varphi_1 = \square\diamond\pi$, while the invariant property can be used to model collision avoidance requirements $\psi = \square\neg\text{obstacle}$. We allow violations of the invariant constraint ψ. To minimize the impact of invariant constraint violations, we investigate the problem of minimizing the invariant constraint violation rate. In particular, given a specification $\varphi = \varphi_1 \wedge \psi$, the objective is to compute a control policy that minimizes the expected number of violations of ψ per cycle over all the stationary policies that maximizes the probability of satisfying φ_1. We say that every visit to a state that satisfies φ_1 completes a *cycle*. The problem investigated in this chapter is formally stated as follows:

Problem 5.1. *Compute a secure control policy μ that minimizes the violation rate of ψ, i.e., the expected number of violations of ψ per cycle, while maximizing the probability that ϕ_1 is satisfied under any adversary policy τ.*

We denote the probability of satisfying LTL specification φ as *satisfaction probability*. When the initial state is given, we use $\mathbb{P}^{\mu_\tau}(\varphi|s)$ to denote the satisfaction probability when starting from state $s \in S$. Problem 4.1 can be found useful in various applications.

To investigate the problem above, we assign a positive cost α to every transition initiated from a state s if $s \not\models \psi$. If state $s \models \psi$, we let $g(s) = 0$ for all u_C and u_A. Thus we have for all u_C and u_A

$$g(s) = \begin{cases} \alpha & \text{if } s \not\models \psi \\ 0 & \text{if } s \models \psi. \end{cases} \tag{5.1}$$

By Proposition 4.1, we have that two consecutive visits to the set of accepting states $\mathcal{E}$ corresponding to φ_1 complete a cycle. Based on the transition cost defined in Eqn. (5.1), Problem 5.1 can be rewritten as follows.

Problem 5.2. *Given a stochastic game $\mathcal{G}$ and an LTL formula φ in the form of $\varphi = \varphi_1 \wedge \psi$, compute an optimal control policy μ that maximizes the probability of satisfying φ_1 while minimizing the average cost per cycle due to violating ψ which is defined as*

$$J^{\mu_\tau} = \limsup_{N\to\infty} \mathbb{E}\left\{ \frac{\sum_{k=0}^{N} g(s_k)}{C(N)} \;\middle|\; \eta_\mu \models \varphi_1 \right\}, \tag{5.2}$$

where $C(N)$ represents the number of cycles completed up to stage N and η_μ is the word generated by the path on SG when the controller takes policy μ.

5.3 Control Synthesis to Minimize Invariant Constraint Violation Rate

This section presents the proposed solution approach. Note that our objective is two-fold: (i) maximize the probability of satisfying φ_1, and (ii) minimize the violation rate of ψ. To achieve (i), we follow the product SG construction proposed in Chapter 4, i.e., construct a product SG $\mathcal{G}_{prod}$ using SG $\mathcal{G}$ and the DRA constructed using specification φ_1. To achieve (ii), we generalize the average reward per stage problem in Chapter 2 to incorporate the presence of the adversary, and derive the optimality condition by bridging Problem 5.2 with the generalized average reward per stage problem. We propose a policy iteration algorithm to compute the control policy leveraging the optimality condition.

Since φ_1 is required to be satisfied, similar to our analysis in Chapter 4, we first construct a product SG $\mathcal{G}_{prod} = (S_{prod}, U_C, U_A, P_{prod}, \text{Acc}_{prod})$ using SG $\mathcal{G}$ and the DRA constructed using specification φ_1. Then we have the following observations. First, the one-to-one correspondence relationships between the control policies, paths, and associated expected cost due to violating ψ on $\mathcal{G}_{prod}$ and $\mathcal{G}$ hold. Furthermore, we observe that if there exists a control policy such that the specification φ can be satisfied, it is the optimal solution to Problem 4.1 in Chapter 4 with $J^{\mu_\tau} = 0$. Finally, by our analysis in Chapter 4, specification φ_1 is guaranteed to be satisfied if there exists a control policy that can reach the set of

accepting states $\mathcal{E}$. These observations provide us the advantage to analyze Problem 5.1 on the product SG $\mathcal{G}_{prod}$ constructed using SG $\mathcal{G}$ and the DRA constructed using φ_1. Hence, in the following, we analyze Problem 5.1 on the product SG $\mathcal{G}_{prod}$. When the context is clear, we use s to refer to state $(s, q) \in S_{prod}$. Without loss of generality, we assume that $\mathcal{E} = \{1, 2, \cdots, l\}$, i.e., states $\{l+1, \cdots, n\} \cap \mathcal{E} = \emptyset$.

5.3.1 Optimality Condition Derivation

In this subsection, we derive the optimality condition for minimizing the invariant constraint violation rate. For an SG, the average reward per stage problem is to maximize

$$B^{\mu\tau}(s) = \limsup_{N\to\infty} \frac{1}{N}\mathbb{E}\left\{\sum_{n=0}^{N} -g(s) \mid s_0 = s\right\} \tag{5.3}$$

over all stationary control policies considering the adversary plays some strategy τ against the controller. Since we consider a cost is incurred at each stage, we say $B^{\mu\tau}$ is the average cost per stage.

Given any stationary policies μ and τ, denote the induced transition probability matrix as $P^{\mu\tau}$ with $P^{\mu\tau}(s'|s) = \sum_{u_C \in U_C(s)} \sum_{u_A \in U_A(s)} \mu(s, u_C)\tau(s, u_A)P_{prod}(s'|s, u_C, u_A)$. Analogously, denote the expected transition cost starting from any state $s \in S_{prod}$ as $g^{\mu\tau}(s) = \sum_{u_C \in U_C(s)} \sum_{u_A \in U_A(s)} \mu(s, u_C)\tau(s, u_A)g(s)$. Similar to [101], a gain-bias pair is used to characterize the optimality condition. The gain-bias pair $(B^{\mu\tau}, b^{\mu\tau})$ under stationary policies μ and τ satisfies the following proposition, where $B^{\mu\tau}$ is the average cost per stage and $b^{\mu\tau}$ is the differential or relative cost vector.

Lemma 5.1. *Let μ and τ be proper stationary policies for a communicating SG, where a communicating SG is an SG whose underlying graph is strongly connected. Then there exists a constant $\zeta^{\mu\tau}$ such that*

$$B^{\mu\tau}(s) = \zeta^{\mu\tau}, \ \forall s \in S_{prod}. \tag{5.4}$$

Furthermore, the gain-bias pair $(B^{\mu\tau}, b^{\mu\tau})$ satisfies

$$B^{\mu\tau}(s) + b^{\mu\tau}(s) = g^{\mu\tau}(s) + \sum_{k=1}^{n} P^{\mu\tau}(k|s)b^{\mu\tau}(k) \tag{5.5}$$

Proof. Suppose s' is a recurrent state under policies μ and τ. Define $\xi(s)$ as the expected cost to reach s' for the first time from state s, and $o(s)$ as the expected number of stages to reach s' for the first time from s. Thus $\xi(s')$ and $o(s')$ can be interpretated as the expected cost and expected number of stages to return to s' for the first time from state s', respectively.

Based on the definitions above, we have the following equations:

$$\xi(s) = g^{\mu\tau}(s) + \sum_{k \in S_{prod} \setminus s'} P^{\mu\tau}(k|s)\xi(k), \ \forall s \in S_{prod}, \tag{5.6}$$

$$o(s) = 1 + \sum_{k \in S_{prod} \setminus s'} P^{\mu\tau}(k|s)o(k), \ \forall s \in S_{prod}. \tag{5.7}$$

Define $\zeta^{\mu\tau} = \xi(s')/o(s')$. Multiplying Eqn. (5.7) by $\zeta^{\mu\tau}$ and subtracting the associated product from Eqn. (5.6), we have

$$\xi(s) - \zeta^{\mu\tau} o(s) = g^{\mu\tau}(s) - \zeta^{\mu\tau} + \sum_{k=S_{prod} \setminus s'}^{n} P^{\mu\tau}(k|s)(\xi(k) - \zeta^{\mu\tau} o(k)), \ \forall s \in S_{prod}. \tag{5.8}$$

Define a bias term

$$b^{\mu\tau}(s) = \xi(s) - \zeta^{\mu\tau} o(s), \ \forall s \in S_{prod} \tag{5.9}$$

Using Eqn. (5.9), Eqn. (5.8) can be rewritten as

$$\zeta^{\mu\tau} + b^{\mu\tau}(s) = g^{\mu\tau}(s) + \sum_{k=1}^{n} P^{\mu\tau}(k|s) b^{\mu\tau}(k), \ \forall s \in S_{prod} \tag{5.10}$$

which completes our proof. □

The result presented above generalizes the one in [101] in the sense that we consider the presence of adversary. The reason that we focus on communicating SG is that we will focus on the accepting states which are strongly connected. Based on Lemma 5.1, we have the optimality conditions for generalized average cost per stage problem expressed using the gain-bias pair (B, b):

$$B(s) = \min_{\mu} \max_{\tau} \sum_{u_C \in U_C} \sum_{u_A \in U_A} \sum_{s'} \mu(s, u_C)\tau(s, u_A) P_{prod}(s'|s, u_C, u_A) B(s') \tag{5.11}$$

$$B(s) + b(s) = \min_{\mu \in \mu^*} \max_{\tau \in \tau^*} \left[g^{\mu\tau}(s) + \sum_{u_C \in U_C} \sum_{u_A \in U_A} \sum_{s'} \mu(s, u_C)\tau(s, u_A) P_{prod}(s'|s, u_C, u_A) b(s') \right] \tag{5.12}$$

where μ^* and τ^* are the optimal policy sets obtained by solving (5.11). Eqn. (5.11) can be shown using the method presented in Lemma 4.1, and Eqn. (5.12) is obtained directly from (5.10). Given the optimality conditions (5.11) and (5.12) for generalized average cost per stage problem, we can derive the optimality conditions for Problem 5.2 by mapping Problem 5.2 to generalized average cost per stage problem.

We assume that $|S_{prod}| = n$. Denote the gain-bias pair of Problem 5.2 on the product

SG $\mathcal{G}_{prod}$ under policies μ and τ as $(J^{\mu\tau}, h^{\mu\tau})$, where $J^{\mu\tau} = [J^{\mu\tau}(1), J^{\mu\tau}(2), \cdots, J^{\mu\tau}(n)]^\top$ and $h^{\mu\tau} = [h^{\mu\tau}(1), h^{\mu\tau}(2), \cdots, h^{\mu\tau}(n)]^\top$. We can express the transition probability matrix $P^{\mu\tau}$ induced by control and adversary policy μ and τ as $P^{\mu\tau} = P^{\mu\tau}_{\text{in}} + P^{\mu\tau}_{\text{out}}$, where

$$P^{\mu\tau}_{\text{in}}(s'|s) = \begin{cases} P^{\mu\tau}(s'|s) & \text{if } s' \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}, \qquad P^{\mu\tau}_{\text{out}}(s'|s) = \begin{cases} P^{\mu\tau}(s'|s) & \text{if } s' \notin \mathcal{E} \\ 0 & \text{otherwise} \end{cases}. \quad (5.13)$$

Denote the probability that some accepting state $s' \in \mathcal{E}$ is visited from state s under policies μ and τ as $\hat{P}^{\mu\tau}(s'|s)$. Then we see that $\hat{P}^{\mu\tau}(s'|s)$ is calculated as

$$\begin{aligned} \hat{P}^{\mu\tau}(s'|s) = &\sum_{u_C \in U_C(s)} \mu(s, u_C) \sum_{u_A \in U_A(s)} \tau(s, u_A) P_{prod}(s'|s, u_C, u_A) \\ &+ \sum_{u_C \in U_C(s)} \mu(s, u_C) \sum_{u_A \in U_A(s)} \tau(s, u_A) \sum_{k=l+1}^{n} P_{prod}(k|s, u_C, u_A) \hat{P}^{\mu\tau}(s'|k). \quad (5.14) \end{aligned}$$

The intuition behind Eqn. (5.14) is that the probability that s' is the first state to be visited consists of the following two parts. The first term in Eqn. (5.14) describes the probability that the next state is in the set of accepting states $\mathcal{E}$. The second term in Eqn. (5.14) models the probability that before reaching state $s' \in \mathcal{E}$, the next visiting state is $k \notin \mathcal{E}$. Denote the transition probability matrix formed by $\hat{P}(s'|s)$ as $\hat{P}^{\mu\tau}$. Since $P^{\mu\tau}_{\text{out}}$ is substochastic and transient, we have $I - P^{\mu\tau}_{\text{out}}$ is non-singular [107], where I is the identity matrix with proper dimension. Thus $I - P^{\mu\tau}_{\text{out}}$ is invertible. Then using Eqn. (5.13), the transition probability matrix $\hat{P}^{\mu\tau}$ is represented as

$$\hat{P}^{\mu\tau} = (I - P^{\mu\tau}_{\text{out}})^{-1} P^{\mu\tau}_{\text{in}}. \quad (5.15)$$

Denote the expected invariant property violation cost incurred when visiting some accepting state $s' \in \mathcal{E}$ from state s under policies μ and τ as $\hat{g}(s)$. The expected cost $\hat{g}(s)$ is calculated as follows:

$$\hat{g}(s) = g^{\mu\tau}(s) + \sum_{k=l+1}^{n} P^{\mu\tau}(k|s) \hat{g}(k). \quad (5.16)$$

We denote the expected cost vector formed by $\hat{g}(s)$ under policies μ and τ as $\hat{g}^{\mu\tau}$. Then using Eqn. (5.13), the expected cost vector (5.16) can be rearranged as follows:

$$\hat{g}^{\mu\tau} = P^{\mu\tau}_{\text{out}} \hat{g}^{\mu\tau} + g^{\mu\tau} \Rightarrow \hat{g}^{\mu\tau} = (I - P^{\mu\tau}_{\text{out}})^{-1} g^{\mu\tau}. \quad (5.17)$$

Using Eqn. (5.15) and (5.17), we can rewrite (5.2) as

$$J^{\mu\tau} = \limsup_{N \to \infty} \frac{1}{N} \sum_{k=0}^{N-1} \hat{P}^{\mu\tau^k} \hat{g}^{\mu\tau}. \quad (5.18)$$

Policies μ and τ of the product SG $\mathcal{G}_{prod}$ for Problem 5.2 are related to proper policies $\hat{\mu}$ and $\hat{\tau}$ for generalized average cost per stage problem as follows:

$$\hat{P}^{\mu\tau} = P^{\hat{\mu}\hat{\tau}}, \quad \hat{g}^{\mu\tau} = g^{\hat{\mu}\hat{\tau}}, \quad J^{\mu\tau} = B^{\hat{\mu}\hat{\tau}}. \tag{5.19}$$

If we define a bias term $h^{\mu\tau} = b^{\hat{\mu}\hat{\tau}}$, then a gain-bias pair $(J^{\mu\tau}, h^{\mu\tau})$ is constructed for Problem 5.2. Under the worst-case adversary policy $\hat{\tau}$, the control policy that makes the gain-bias pair of average cost per stage problem satisfy

$$B + b \leq g^{\hat{\mu}\hat{\tau}} + P^{\hat{\mu}\hat{\tau}} b \tag{5.20}$$

is optimal. That is, the control policy μ^* that maps to $\hat{\mu}^*$ is optimal.

To obtain the optimal control policy, we need to characterize Problem 5.2 in terms of the control and adversary policies μ and τ. The following lemma generalizes the results presented in [26] in which no adversary is considered. For completeness, we show its proof which generalizes the proof in [26].

Lemma 5.2. *The gain-bias pair $(J^{\mu\tau}, h^{\mu\tau})$ of Problem 5.2 under policies μ and τ satisfies the following equations:*

$$J^{\mu\tau} = P^{\mu\tau} J^{\mu\tau}, \tag{5.21}$$

$$J^{\mu\tau} + h^{\mu\tau} = g^{\mu\tau} + P^{\mu\tau} h^{\mu\tau} + P^{\mu\tau}_{out} J^{\mu\tau}, \tag{5.22}$$

$$P^{\mu\tau} v^{\mu\tau} = (I - P^{\mu\tau}_{out}) h^{\mu\tau} + v^{\mu\tau}, \tag{5.23}$$

for some vector $v^{\mu\tau}$.

Proof. Given the policies $\hat{\mu}$ and $\hat{\tau}$ for average cost per stage problem, we have

$$\begin{aligned} J^{\hat{\mu}\hat{\tau}} &= P^{\hat{\mu}\hat{\tau}} J^{\hat{\mu}\hat{\tau}}, \\ J^{\hat{\mu}\hat{\tau}} + h^{\hat{\mu}\hat{\tau}} &= g^{\hat{\mu}\hat{\tau}} + P^{\hat{\mu}\hat{\tau}} h^{\hat{\mu}\hat{\tau}}, \\ h^{\hat{\mu}\hat{\tau}} + v^{\hat{\mu}\hat{\tau}} &= P^{\hat{\mu}\hat{\tau}} v^{\hat{\mu}\hat{\tau}}, \end{aligned}$$

Due to the connection between the control policy of Problem 5.2 and generalized average cost per stage problem, we have

$$J^{\mu\tau} = P^{\hat{\mu}\hat{\tau}} J^{\mu\tau} = (I - P^{\mu\tau}_{\text{out}})^{-1} P^{\mu\tau}_{\text{in}} J^{\mu\tau}.$$

By rearranging the equation above, we have

$$(I - P^{\mu\tau}_{\text{out}}) J^{\mu\tau} = J^{\mu\tau} - P^{\mu\tau}_{\text{out}} J^{\mu\tau} = P^{\mu\tau}_{\text{in}} J^{\mu\tau}.$$

Thus

$$J^{\mu\tau} = (P^{\mu\tau}_{\text{out}} + P^{\mu\tau}_{\text{in}}) J^{\mu\tau} = P^{\mu\tau} J^{\mu\tau}.$$

The expression $J^{\mu\tau} + h^{\mu\tau} = g^{\mu\tau} + P^{\mu\tau}h^{\mu\tau} + P^{\mu\tau}_{\text{out}}J^{\mu\tau}$ can be rewritten using Eqn. (5.15) and (5.17) as

$$J^{\mu\tau} + h^{\mu\tau} = (I - P^{\mu\tau}_{\text{out}})^{-1}(g^{\mu\tau} + P^{\mu\tau}_{\text{in}}h^{\mu\tau}).$$

Manipulating the equation above, we see that $(I - P^{\mu\tau}_{\text{out}})(J^{\mu\tau} + h^{\mu\tau}) = g^{\mu\tau} + P^{\mu\tau}_{\text{in}}h^{\mu\tau}$. Then we can see that

$$J^{\mu\tau} + h^{\mu\tau} = g^{\mu\tau} + (P^{\mu\tau}_{\text{in}} + P^{\mu\tau}_{\text{out}})h^{\mu\tau} + P^{\mu\tau}_{\text{out}}J^{\mu\tau} = g^{\mu\tau} + P^{\mu\tau}h^{\mu\tau} + P^{\mu\tau}_{\text{out}}J^{\mu\tau}.$$

Start from $h^{\hat{\mu}\hat{\tau}} + v^{\hat{\mu}\hat{\tau}} = P^{\hat{\mu}\hat{\tau}}v^{\hat{\mu}\hat{\tau}}$. We see that $h^{\hat{\mu}\hat{\tau}} + v^{\hat{\mu}\hat{\tau}} = (I - P^{\mu\tau}_{\text{out}})^{-1}P^{\mu\tau}_{\text{in}}v^{\mu\tau}$. Therefore we have

$$(I - P^{\mu\tau}_{\text{out}})h^{\mu\tau} + v^{\mu\tau} = P^{\mu\tau}v^{\mu\tau},$$

which completes our proof. □

Lemma 5.2 indicates that the gain-bias pair can be solved as solutions to a linear system with $3n$ unknowns. Thus we can evaluate any control and adversary policies using Lemma 5.2, which provides us the potential to implement iterative algorithm to compute the optimal control policy μ.

To compute the control policy μ, we define two operators on (J, h), denoted as $T^*(J, h)$ and $T(J, h)$, given as:

$$\begin{aligned}
(T^*(J,h))(s) = \min_{\mu}\max_{\tau}\Bigg[& \sum_{u_C\in U_C(s)}\sum_{u_A\in U_A(s)} \mu(s,u_C)\tau(s,u_A)g(s) \\
& + \sum_{u_C\in U_C(s)}\sum_{u_A\in U_A(s)}\sum_{s'=1}^{n} \mu(s,u_C)\tau(s,u_A)P_{prod}(s'|s,u_C,u_A)h(s') \\
& + \sum_{u_C\in U_C(s)}\sum_{u_A\in U_A(s)}\sum_{s'=l+1}^{n} \mu(s,u_C)\tau(s,u_A)P_{prod}(s'|s,u_C,u_A)J(s')\Bigg],\ \forall s
\end{aligned} \tag{5.24}$$

$$\begin{aligned}
(T_\mu(J,h))(s) = \max_{\tau}\Bigg[& \sum_{u_C\in U_C(s)}\sum_{u_A\in U_A(s)} \mu(s,u_C)\tau(s,u_A)g(s) \\
& + \sum_{u_C\in U_C(s)}\sum_{u_A\in U_A(s)}\sum_{s'=1}^{n} \mu(s,u_C)\tau(s,u_A)P_{prod}(s'|s,u_C,u_A)h(s') \\
& + \sum_{u_C\in U_C(s)}\sum_{u_A\in U_A(s)}\sum_{s'=l+1}^{n} \mu(s,u_C)\tau(s,u_A)P_{prod}(s'|s,u_C,u_A)J(s')\Bigg].\ \forall s
\end{aligned} \tag{5.25}$$

Generally speaking, we can view them as mappings from (J, h) to $T^*(J, h) \in \mathbb{R}^n$ and $T_\mu(J, h) \in \mathbb{R}^n$, respectively. Note that in Eqn. (5.25), the transition probability is the one induced under a certain control policy μ. Based on the definitions of operators $T^*(J, h)$ and $T(J, h)$, we present the optimality conditions for Problem 5.2 using the following theorem.

Theorem 5.1. *The control policy μ with gain-bias pair $(J^{\mu\tau}, h^{\mu\tau})$ that satisfies*

$$J^{\mu\tau} + h^{\mu\tau} = T^*(J^{\mu\tau}, h^{\mu\tau}) \tag{5.26}$$

is the optimal control policy.

Proof. Consider any arbitrary control policy $\hat{\mu}$ and the worst-case adversary policy $\hat{\tau}$. By definition of $T^*(\cdot)$ in Eqn. (5.24), we have that Eqn. (5.26) implies

$$J^{\mu\tau} + h^{\mu\tau} \leq g^{\hat{\mu}\hat{\tau}} + P^{\hat{\mu}\hat{\tau}} h^{\mu\tau} + P^{\hat{\mu}\hat{\tau}}_{\text{out}} J^{\mu\tau},$$

where $P^{\hat{\mu}\hat{\tau}}$ and $P^{\hat{\mu}\hat{\tau}}_{\text{out}}$ are the transition probability matrix induced by policies $\hat{\mu}$ and $\hat{\tau}$. Then we have that

$$J^{\mu\tau} + h^{\mu\tau} - P^{\hat{\mu}\hat{\tau}}_{\text{out}} J^{\mu\tau} \leq g^{\hat{\mu}\hat{\tau}} + P^{\hat{\mu}\hat{\tau}} h^{\mu\tau} = g^{\hat{\mu}\hat{\tau}} + (P^{\hat{\mu}\hat{\tau}}_{\text{in}} + P^{\hat{\mu}\hat{\tau}}_{\text{out}}) h^{\mu\tau}.$$

Thus we observe that

$$(I - P^{\hat{\mu}\hat{\tau}}_{\text{out}})(J^{\mu\tau} + h^{\mu\tau}) \leq g^{\hat{\mu}\hat{\tau}} + P^{\hat{\mu}\hat{\tau}}_{\text{in}} h^{\mu\tau}.$$

Note that $(I - P^{\hat{\mu}\hat{\tau}}_{\text{out}})$ is invertible. Thus the inequality above is rewritten as

$$J^{\mu\tau} + h^{\mu\tau} \leq (I - P^{\hat{\mu}\hat{\tau}}_{\text{out}})^{-1}(g^{\hat{\mu}\hat{\tau}} + P^{\hat{\mu}\hat{\tau}}_{\text{in}} h^{\mu\tau}).$$

Rewrite the inequality above according to Eqn. (5.15) and (5.17). Then we have

$$J^{\mu\tau} + h^{\mu\tau} \leq g^{\tilde{\mu}\tilde{\tau}} + P^{\tilde{\mu}\tilde{\tau}} h^{\mu\tau},$$

where $\tilde{\mu}$ and $\tilde{\tau}$ are the control and adversary policies in the associated average cost per cycle problem. Thus, $\tilde{\mu}^*$ satisfies Eqn. (5.20) and μ is optimal over all the proper policies. □

5.3.2 Policy Iteration Algorithm for Control Synthesis

In the following, we focus on how to obtain an optimal secure control policy. First, note that the optimal control policy consists of two parts. The first part, denoted as μ_{reach}, maximizes the probability of satisfying specification φ_1, while the second part, denoted as μ_{cycle}, minimizes the violation cost per cycle due to violating invariant property ψ. Following the procedure described in Algorithm 7 (see Chapter 4), we can obtain the control policy μ_{reach} that maximizes the probability of satisfying specification φ_1. Suppose the set of accepting states $\mathcal{E}$ has been reached. Then the control policy μ_{cycle} that optimizes the long term performance of the system is generated using Algorithm 8. Algorithm 8 first initializes the control and adversary policies arbitrarily (e.g., if μ^0 and τ^0 are set as uniform distributions, then $\mu^0(s, u_C) = 1/|U_C(s)|$ and $\tau^0(s, u_A) = 1/|U_A(s)|$ for all s, u_C and u_A).

Algorithm 8 Algorithm for a control strategy that minimizes the expected number of invariant constraint violations.

Input: product SG $\mathcal{G}_{prod}$, the set GAMECs $\mathcal{C}$ associated with formula φ_1
Output: Control policy μ_{cycle}
Initialization: Initialize μ^0 and τ^0. Initialize iteration index $k \leftarrow 0$.
while $T^*(J^{\mu^k\tau^k}, h^{\mu^k\tau^k}) \neq T^*(J^{\mu^{k-1}\tau^{k-1}}, h^{\mu^{k-1}\tau^{k-1}})$ **do**
    Policy Evaluation: Given μ^k and τ^k, calculate the gain-bias pair $(J^{\mu^k\tau^k}, h^{\mu^k\tau^k})$ using Lemma 5.2.
    Policy Improvement: Calculate the control policy μ using $\mu = \arg\min_\mu \arg\max_\tau \left\{ g^{\mu\tau} + P^{\mu\tau} h^{\mu^k\tau^k} + P^{\mu\tau}_{\text{out}} J^{\mu^k\tau^k} \right\}$.
    $\mu^{k+1} \leftarrow \mu$.
    $k \leftarrow k + 1$.
end while

Then it follows a policy iteration procedure to update the control and the corresponding adversary policies until no more improvement can be made. Given μ_{reach} and μ_{cycle}, we can construct the optimal control policy for Problem 5.2 as

$$\mu^* = \begin{cases} \mu_{\text{reach}}, & \text{if } s \notin \mathcal{E} \\ \mu_{\text{cycle}}, & \text{if } s \in \mathcal{E} \end{cases} . \tag{5.27}$$

We finally present the convergence and optimality of Algorithm 8 using the following theorem.

Theorem 5.2. *Algorithm 8 terminates within a finite number of iterations for any given accepting state set $\mathcal{E}$. Moreover, the result returned by Algorithm 8 satisfies the optimality conditions for Problem 5.2.*

Proof. In the following, we first prove Algorithm 8 converges within a finite number of iterations. Then we prove that the results returned by Algorithm 8 satisfies the optimality conditions in Theorem 5.1. We denote the iteration index as k. The control policy at k-th iteration is denoted as μ^k. The worst-case adversary policy associated with μ^k is denoted as τ^k. Let $\mathbf{1}$ be a vector with all entries equal to one with appropriate size. Define a vector $\varepsilon \in \mathbb{R}^n$ as

$$\varepsilon = J^{\mu^k\tau^k}\mathbf{1} + h^{\mu^k\tau^k} - g^{\mu^{k+1}\tau^{k+1}} - P^{\mu^{k+1}\tau^{k+1}} h^{\mu^k\tau^k} - P^{\mu^{k+1}\tau^{k+1}}_{\text{out}} J^{\mu^k\tau^k}\mathbf{1}.$$

By Lemma 5.2, we have that

$$J^{\mu^k\tau^k}\mathbf{1} + h^{\mu^k\tau^k} = g^{\mu\tau} + P^{\mu\tau} h^{\mu^k\tau^k} + P^{\mu\tau}_{\text{out}} J^{\mu^k\tau^k}.$$

By the definition of $T^*(\cdot)$ in Eqn. (5.24), the control policy at iteration $k+1$ is computed by optimizing $g^{\mu\tau} + P^{\mu\tau} h^{\mu^k\tau^k} + P^{\mu\tau}_{\text{out}} J^{\mu^k\tau^k}$. Thus we have that for all s, $\varepsilon(s) \geq 0$. Moreover,

we can rewrite vector ε as

$$\begin{aligned}\varepsilon &= J^{\mu^k\tau^k}\mathbf{1} + h^{\mu^k\tau^k} - g^{\mu^{k+1}\tau^{k+1}} - P^{\mu^{k+1}\tau^{k+1}}h^{\mu^{k+1}\tau^{k+1}} - P_{\text{out}}^{\mu^{k+1}\tau^{k+1}}J^{\mu^{k+1}\tau^{k+1}}\mathbf{1} \\ &\quad + P^{\mu^{k+1}\tau^{k+1}}h^{\mu^{k+1}\tau^{k+1}} + P_{\text{out}}^{\mu^{k+1}\tau^{k+1}}J^{\mu^{k+1}\tau^{k+1}}\mathbf{1} - P^{\mu^{k+1}\tau^{k+1}}h^{\mu^k\tau^k} - P_{\text{out}}^{\mu^{k+1}\tau^{k+1}}J^{\mu^k\tau^k}\mathbf{1} \\ &= J^{\mu^k\tau^k}\mathbf{1} + h^{\mu^k\tau^k} - J^{\mu^{k+1}\tau^{k+1}}\mathbf{1} - h^{\mu^{k+1}\tau^{k+1}} - P^{\mu^{k+1}\tau^{k+1}}\left(h^{\mu^k\tau^k} - h^{\mu^{k+1}\tau^{k+1}}\right) \\ &\quad - P_{\text{out}}^{\mu^{k+1}\tau^{k+1}}\left(J^{\mu^k\tau^k} - J^{\mu^{k+1}\tau^{k+1}}\right)\mathbf{1},\end{aligned}$$

where the second equality holds by Lemma 5.2. Thus ε can be represented as

$$\varepsilon = \left(I - P_{\text{out}}^{\mu^{k+1}\tau^{k+1}}\right)\left(J^{\mu^k\tau^k} - J^{\mu^{k+1}\tau^{k+1}}\right)\mathbf{1} + \left(I - P^{\mu^{k+1}\tau^{k+1}}\right)\left(h^{\mu^k\tau^k} - h^{\mu^{k+1}\tau^{k+1}}\right), \tag{5.28}$$

where I is the identity matrix. By multiplying $\left(P^{\mu^{k+1}\tau^{k+1}}\right)^t$ to both sides of Eqn. (5.28) and calculating the summation over t from 0 to $T-1$, we have that

$$\begin{aligned}\sum_{t=0}^{T-1}\left(P^{\mu^{k+1}\tau^{k+1}}\right)^t\varepsilon &= \sum_{t=0}^{T-1}\left(P^{\mu^{k+1}\tau^{k+1}}\right)^t\left(I - P_{\text{out}}^{\mu^{k+1}\tau^{k+1}}\right)\left(J^{\mu^k\tau^k} - J^{\mu^{k+1}\tau^{k+1}}\right)\mathbf{1} \\ &\quad + \sum_{t=0}^{T-1}\left(P^{\mu^{k+1}\tau^{k+1}}\right)^t\left(I - P^{\mu^{k+1}\tau^{k+1}}\right)\left(h^{\mu^k\tau^k} - h^{\mu^{k+1}\tau^{k+1}}\right).\end{aligned} \tag{5.29}$$

Divide both sides by T and let $T \to \infty$. Then we have

$$\begin{aligned}&\lim_{T\to\infty}\sum_{t=0}^{T-1}\frac{1}{T}\left(P^{\mu^{k+1}\tau^{k+1}}\right)^t\varepsilon \\ &\quad = \lim_{T\to\infty}\sum_{t=0}^{T-1}\frac{1}{T}\Big(\left(P^{\mu^{k+1}\tau^{k+1}}\right)^t\left(P^{\mu^{k+1}\tau^{k+1}}\right)^t P_{\text{out}}^{\mu^{k+1}\tau^{k+1}}\Big)\big(J^{\mu^k\tau^k} - J^{\mu^{k+1}\tau^{k+1}}\big)\mathbf{1}\end{aligned} \tag{5.30}$$

since the second term of (5.29) is eliminated when $T \to \infty$. Note that $P_{\text{out}}^{\mu^{k+1}\tau^{k+1}}$ is a substochastic matrix. We have that $P_{\text{out}}^{\mu^{k+1}\tau^{k+1}}\mathbf{1} \leq \mathbf{1}$. Furthermore, since $P^{\mu^{k+1}\tau^{k+1}}$ is a stochastic matrix, we see that $\mathbf{1} - P_{\text{out}}^{\mu^{k+1}\tau^{k+1}}\mathbf{1} \geq 0$. Thus we have

$$\left(\left(P^{\mu^{k+1}\tau^{k+1}}\right)^t - \left(P^{\mu^{k+1}\tau^{k+1}}\right)^t P_{\text{out}}^{\mu^{k+1}\tau^{k+1}}\right)\mathbf{1} \geq 0.$$

Given the inequality above and $\varepsilon \geq 0$, we have that $J^{\mu^k\tau^k} - J^{\mu^{k+1}\tau^{k+1}} \geq 0$ by observing (5.30), which implies that $J^{\mu^k\tau^k} \geq J^{\mu^{k+1}\tau^{k+1}}$.

Consider the scenario where $J^{\mu^k\tau^k} = J^{\mu^{k+1}\tau^{k+1}}$. We further need to show that in this case

$h^{\mu^k\tau^k} \geq h^{\mu^{k+1}\tau^{k+1}}$. For each state that belongs to the recurrent class, the corresponding entry of $\sum_{t=0}^{T-1}\left(P^{\mu^{k+1}\tau^{k+1}}\right)^t$ is positive. By observing Eqn. (5.30), we have $\varepsilon(s) = 0$ for all s belonging to the recurrent class. Thus according to Eqn. (5.29), we have that $h^{\mu^k\tau^k}(s) = h^{\mu^{k+1}\tau^{k+1}}(s)$ for all s in the recurrent class.

By Eqn. (5.29), we have that

$$\begin{aligned}\lim_{T\to\infty}\sum_{t=0}^{T-1}\left(P^{\mu^{k+1}\tau^{k+1}}\right)^t(h^{\mu^k\tau^k} - h^{\mu^{k+1}\tau^{k+1}}) =& h^{\mu^k\tau^k} - h^{\mu^{k+1}\tau^{k+1}} - \lim_{T\to\infty}\sum_{t=0}^{T-1}\left(P^{\mu^{k+1}\tau^{k+1}}\right)^t\varepsilon \\ \leq& h^{\mu^k\tau^k} - h^{\mu^{k+1}\tau^{k+1}} - \varepsilon.\end{aligned}$$

Note that the elements corresponding to the transient states in $\left(P^{\mu^{k+1}\tau^{k+1}}\right)^t(h^{\mu^k\tau^k} - h^{\mu^{k+1}\tau^{k+1}})$ approach zero when $t \to \infty$. Thus we have $h^{\mu^k\tau^k}(s) - h^{\mu^{k+1}\tau^{k+1}}(s) \geq \varepsilon(s) \geq 0$ for all transient states s. Combining all the above together, we have that $\mu^k = \mu^{k+1}$ when $\delta = 0$, otherwise $h^{\mu^k\tau^k}(s) - h^{\mu^{k+1}\tau^{k+1}}(s) \geq 0$ holds for some transient state s.

When Algorithm 8 terminates, we have that

$$T^*(J^{\mu^{k+1}\tau^{k+1}}, h^{\mu^{k+1}\tau^{k+1}}) = T^*(J^{\mu^k\tau^k}, h^{\mu^k\tau^k}). \tag{5.31}$$

By Algorithm 8, the gain-bias pair $(J^{\mu^k\tau^k}, h^{\mu^k\tau^k})$ is first evaluated using Lemma 5.2 at each iteration k. Then using the gain-bias pair obtained in policy evaluation phase, the T^* operator is calculated as shown in Algorithm 8. Thus according to Lemma 5.2, we see

$$\mu = \arg\min_{\mu}\max_{\tau}\left\{g^{\mu\tau} + P^{\mu\tau}h^{\mu^k\tau^k} + P^{\mu\tau}_{\text{out}}J^{\mu^k\tau^k}\right\}. \tag{5.32}$$

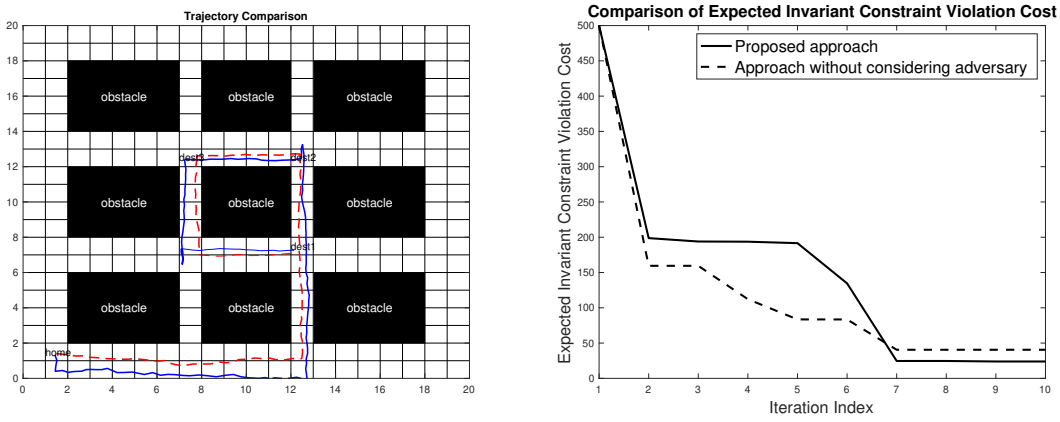
Note that the right hand side of Eqn. (5.32) is equivalent to how T^* is calculated in Algorithm 8. Therefore, by combining Eqn. (5.31) and (5.32), we obtain that

$$J^{\mu^k\tau^k} + h^{\mu^k\tau^k} = T^*(J^{\mu^k\tau^k}, h^{\mu^k\tau^k}).$$

By Theorem 5.1, we see that μ^k is the optimal control policy. □

5.4 Case Study

In this section, we evaluate our proposed approach using a case study on the application of remotely controlled UAV performing a reach-avoid task in a discrete bounded grid environment. The UAV is given an LTL specification $\varphi = \square(\diamond(\text{dest1} \wedge \diamond(\text{dest2} \wedge \diamond\text{dest3}))) \wedge \square\neg\text{obstacle}$ consisting of liveness and invariant constraints. In particular, the lieveness constraint $\varphi_1 = \square(\diamond(\text{dest1} \wedge \diamond(\text{dest2} \wedge \diamond\text{dest3})))$ models a surveillance task, i.e., the UAV is required to patrol three critical regions infinitely often following a particular order, and the



(a) (b)

Figure 5-1: Comparison of the proposed approach and the approach without considering the presence of the adversary. Fig. 5-1a gives the trajectories obtained using two approaches. The solid blue line is the trajectory obtained using the proposed approach, while the dashed red line represents the trajectory obtained using the approach without considering the presence of the adversary. Fig. 5-1b shows the expected invariant constraint violation cost with respect to iteration indices.

invariant constraint $\psi = \square\neg\text{obstacle}$ requires the UAV to avoid collisions with obstacles. Once the critical regions are visited, a cycle is completed. During each cycle, the rate of invariant constraint violation need to be minimized. The cost incurred at each violation is assigned to be 20.

We compare the proposed approach with the approach without considering the adversary. The sample trajectories obtained using these approaches are presented in Fig. 5-1a. In particular, the solid line shows a sample trajectory obtained by using the proposed approach, and the dashed line shows the trajectory obtained by using the control policy synthesized without considering the presence of adversary. We observe that the control strategy synthesized using the approach without considering the adversary uses less effort comparing to the proposed approach. However, the proposed approach is more resilient since it uses more control effort to deviate from the obstacles to minimize the violation cost. We present the average invariant constraint violation cost incurred using the control policy obtained at each iteration in Fig. 5-1b. We observe that the proposed approach incurs lower cost after convergence. In Fig. 5-1b, the approach that does not consider the adversary incurs lower cost compared to the proposed approach during iterations 2 to 6. The reason is that although the proposed approach guarantees convergence to Stackelberg equilibrium, it does not guarantee optimality of the intermediate policies. The average invariant constraint violation cost using proposed approach is 23.90, while the average invariant constraint violation cost using the

approach without considering the adversary is 40.52. The improvement achieved using the proposed approach is 28.67%.

Given the transition probability, the SG and DRA associated with specification ϕ are created within 1 and 0.01 second, respectively. The computation of product SG took 72 seconds. The product SG has 1600 states and 26688 transitions. It took 36 seconds to compute the control policy on a Macbook Pro with 2.6GHz Intel Core i5 CPU and 8GB RAM.

5.5 Conclusion

In this chapter, we investigated the problem of synthesizing a control policy for CPS under malicious attack so as to minimize the invariant constraint violation rate while maximize the probability of satisfying an LTL constraint. We formulated the interaction between the controller and adversary as a concurrent Stackelberg game. We derived the optimality condition of the problem of interest by connecting it with a generalized average cost per stage problem. We developed a policy iteration algorithm to compute an optimal control policy by exploiting the optimality condition. The proposed approach is evaluated using a case study on a UAV performing surveillance task.

Chapter 6

Optimal Minimum Violation Control under Multiple LTL Constraints

6.1 Introduction

Chapter 4 and Chapter 5 focus on CPS subject to one LTL constraint. Currently, CPS are expected to perform under multiple LTL constraints due to their emerging capabilities. When the CPS are subject to multiple LTL constraints, we are faced with two challenges when synthesizing controllers for CPS: (i) the specifications are incompatible and may conflict with each other, and (ii) uncertainties, stochastic errors, and malicious attacks can curtail the satisfaction of the specifications. Both challenges may lead to unsynthesizable controllers. Additionally, we assume that the adversary has perfect observation over the policy taken by the CPS in Chapter 4 and Chapter 5, which may not hold in practice. This motivates us to consider the scenario where the adversary has limited observation over the controller's policy.

In this chapter, we investigate minimum violation control synthesis for a finite-state stochastic game (SG) which serves as an abstraction of CPS under attacks. The players involved in the SG are the controller and adversary. The controller and adversary interact with each other following a concurrent Stackelberg information pattern, in which the controller is the leader and the adversary is the follower. At each state in the SG, both players must take actions simultaneously. The controller is given a set of specifications modeled in co-safe LTL (scLTL). We focus on the scenario where the specifications cannot be completely satisfied, and synthesize the control strategy that minimizes the violations on specifications. The objective of the controller is to maximize the expected reward it obtains when satisfying the specifications, while the objective of the adversary is to compute a strategy to deviate the system from satisfying the specifications, given the control strategy observed by the adversary. To summarize, we make the following contributions:

- A concurrent Stackelberg SG is formulated to model the interaction between the con-

troller and adversary. We use a linear anchoring bias model [108] to model the adversary's limited capability of observation [109]. A real-world application on patrolling security game is presented to demonstrate the effectiveness of the proposed framework.

- The scLTL specifications are expressed using complete and deterministic finite automata. We compute a product SG using the SG and product of the set of automata. A mixed integer nonlinear program (MINLP) is formulated on the product SG to solve for a control strategy that minimally violates specifications.

- Two algorithms are presented to solve the MINLP. We first give an exact algorithm which is based on model reference adaptive search (MRAS) method. We show that MRAS method can be used to solve the proposed MINLP. With probability one, the exact algorithm returns the optimal control policy, regardless of the initial input of the algorithm. We then give an approximate algorithm. The approximate algorithm returns a sub-optimal solution, with a faster convergence rate compared to the exact algorithm.

- We present a numerical case study on patrolling security game to evaluate the proposed framework. The proposed approaches are compared with two baselines, which attempt greedily to satisfy the given specifications. The results show that the proposed framework outperforms the baselines.

The remainder of this chapter is organized as follows. Section 6.2 presents problem formulation and an example on patrolling security game. We formulate an MINLP in Section 6.3, and give two algorithms including an exact algorithm and an approximate algorithm to solve the MINLP. A numerical case study is presented in Section 6.4. We conclude this chapter in Section 6.5 and give a brief discussion on future work.

6.2 System Model and Problem Statement

In this section, we introduce the system model and the problem statement. We also present a real-world security application that can be captured using the problem formulation investigated in this chapter.

We consider an SG $\mathcal{G} = (S, U_C, U_A, P, \Pi, \mathcal{L})$ as defined in Chapter 4 and 5. The SG can be viewed as an abstraction of a CPS in the presence of a malicious adversary. The state transitions of the SG are jointly determined by the actions taken by the controller and adversary.

Note that we consider the Stackelberg setting between the controller and adversary. The controller commits to a control strategy μ first. Then the adversary observes the strategy committed by the controller and responds to its observed strategy based on its own interest. During this interaction process, several restrictions are imposed on both players. First, once the controller commits to some strategy μ, it has no chance to adjust the committed strategy.

Second, the adversary may have limited observation capability. In this chapter, we consider the linear anchoring bias model [108], in which the observation that the adversary perceives is computed as

$$\tilde{\mu}(s, u_C) = \lambda \frac{1}{|U_C(s)|} + (1 - \lambda)\mu(s, u_C), \ \forall s, u_C \tag{6.1}$$

where $\lambda \in [0, 1]$ is a parameter that is known to the controller. Model (6.1) indicates that the adversary observes the strategy committed by the controller $\tilde{\mu}$, which is a linear combination of a uniform distribution and the real control strategy at each state. Thus the control strategy observed by the adversary does not necessarily equal to the control strategy μ. We remark that the adversary does not know it is biased. The last restriction is that at each state both the controller and adversary must take actions simultaneously, i.e., we consider the concurrent Stackelberg setting. Under the concurrent setting, the players do not know the realized actions that being taken by their opponents.

Let $\Phi = \{\phi_1, \phi_2, \cdots, \phi_n\}$ be a set of scLTL specifications. The controller assigns each specification ϕ_i a reward $r(\phi_i)$, based on the specification's importance. We assume that the reward associated with each specification is known to the adversary. The objective of the controller is to compute a control strategy μ on SG $\mathcal{G}$ to maximize its expected total reward. In contrast, the objective of the adversary is to compute a strategy τ to deviate the system from satisfying the specifications in Φ, given its observation over the control strategy μ. Considering the potential incompatibility among the specifications in Φ [58] and the presence of adversary, violations of the specifications need to be considered. We formally state the minimum violation problem on SG as follows:

Problem 6.1. *Given an SG $\mathcal{G}$ under concurrent Stackelberg setting and a set of specifications $\Phi = \{\phi_1, \cdots, \phi_n\}$ modeled in scLTL, with each $\phi_i \in \Phi$ associated with a reward function $r(\phi_i)$, compute a control policy μ such that control policy μ and adversary policy $\tau \in \mathcal{BR}(\tilde{\mu})$ constitute Stackelberg equilibrium of game $\mathcal{G}$, where $\tilde{\mu}$ is the policy observed by the adversary as defined by the anchoring bias model* (6.1).

In the remainder of this section, we present a real-world security application that can be formulated as Problem 6.1.

Security Game with one type of adversary and execution uncertainty: Security games such as green security game (GSG) [110] and patrolling security game (PSG) [51, 111] are used for protection of wild life and critical infrastructure. In the following, we take PSG as an example, and show how PSG is captured by the problem of interest.

A PSG consists of two players, denoted as the defender and adversary. The environment is discretized into a finite set of cells $N = \{n_1, \cdots, n_K\}$. The defender has a finite set of patrol units, e.g., mobile robots or uniformed patrol force. Each patrol unit can move among the cells in N following some schedule determined by the defender. A schedule specified by the defender is a sequence of commands, with each command indicating the patrol unit should reach some cell n at time t. The adversary has control over a finite set of intruders.

Each intruder can intrude in some cell at a certain time t following the instruction from the adversary.

When the patrol unit navigates in the environment following the commands of the defender, there exists execution uncertainty, and hence the transition of each patrol unit between cells is stochastic [112]. For example, the patrol unit might deviate from the prespecified patrol schedule due to writing citations, felony arrest, and handling emergency situations. Such stochastic behavior is captured by a transition probability distribution between the cells, given the commands issued by the defender and adversary.

To show how PSG is modeled using the framework in this chapter, we give the following mappings when there exist a single patrol unit and one intruder. The controller in SG $\mathcal{G}$ is the defender, and the adversary is that of the PSG. The set of states S of SG is obtained as $S = N \times N \times T$, where T is the finite set of discrete time instances. The Cartesian product $N \times N$ captures the joint locations of the patrol unit and intruder. The extension to multiple patrol units and intruders can be obtained by taking the Cartesian product of the locations among all patrol units and intruders, so that each state captures the joint locations of all patrol units and intruders at each time $t \in T$. The finite set of control actions U_C is the set of commands issued by the defender. The finite set of adversary actions U_A is the set of intrusion actions of the intruder. The transition probability P captures the execution uncertainty. For example, when the defender moves the patrol unit from location n to n' at time t' and the adversary initiates no intrusion, the probability of transiting from (n, t) to (n', t') is then given as $P((n', t')|(n, t), u_C, u_A)$.

Some typical specifications that can be given to the defender include: the reach-avoid specification (eventually reach the target region and avoid the obstacles, i.e., $\neg obstacle \mathrm{U} target$), and eventually capture the intruder ($\diamond capture\_intruder$). We finally remark that since time index is incorporated in the state variable, time constrained specifications can also be given to the defender.

6.3 Proposed Approach for Minimum Violation Control Synthesis

In this section, we present the solution approach to Problem 6.1. We first construct a product SG using SG $\mathcal{G}$ and the set of DFAs representing the scLTL formulas in Φ. We then formulate a mixed integer non-linear program (MINLP) on the product SG to compute the Stackelberg equilibrium. We propose two algorithms to solve the problem. The first algorithm, denoted as exact algorithm, is based on model reference adaptive search (MRAS) method. With probability one the exact algorithm returns the optimal control policy. However, it has relatively slow convergence rate. The second algorithm is a heuristic based approximate algorithm, which returns a sub-optimal solution with faster convergence rate.

6.3.1 Product SG Construction

Given the set of specifications Φ, we first construct a set of DFAs $\mathcal{A} = \{A_1, \cdots, A_n\}$ as defined in Chapter 3.4, with each $A_i = (Q^i, q_0^i, \Sigma, \delta^i, F^i)$. To capture the violations of the specifications, we make all DFAs in $\mathcal{A}$ *complete*. A DFA A is said to be complete if transition $\delta(q, \sigma)$ is well-defined on A for all $q \in Q$ and $\sigma \in \Sigma$. An incomplete DFA can be made complete by the following procedure. We first augment the state set with an additional absorbing state, denoted as *sink*, and let $\delta(q, \sigma) = sink$ if $\delta(q, \sigma)$ is not defined in A. Note that the completion procedure does not change the acceptance condition of each DFA A_i. In the remainder of this chapter, with a slight abuse of notations, we use A_i to denote the complete DFA associated with specification ϕ_i and $\mathcal{A}$ to denote the set of complete DFAs.

Given the set of complete DFAs $\mathcal{A}$, we construct a product automaton defined as follows.

Definition 6.1. (Product automaton): *A product automaton obtained from $\mathcal{A}$ is a tuple $A_{prod} = (Q_{prod}, q_{0,prod}, \Sigma, \delta_{prod}, F_{prod})$, where $Q_{prod} = Q^1 \times \cdots \times Q^n$ is a finite set of states, $q_{0,prod} = (q_0^1, \cdots, q_0^n)$ is the initial state, Σ is the alphabet inherited from $\mathcal{A}$, $\delta_{prod} = ((q^1, \cdots, q^n), \sigma, (\hat{q}^1, \cdots, \hat{q}^n))$ if $\delta^i(q^i, \sigma) = \hat{q}^i$ for all i and $F_{prod} = \{(q^1, \cdots, q^n) | q^i \in F^i, \forall i\}$.*

Note that since each DFA A_i is complete, the product automaton A_{prod} is also complete, which allows us to investigate violations of any subset of specifications in 2^Φ.

Given the product automaton, we then construct a product SG using SG $\mathcal{G}$ and product automaton A_{prod} as follows:

Definition 6.2. (Product SG): *Given SG $\mathcal{G} = (S, U_C, U_A, P, \mathcal{L}, \Pi)$ and product automaton $A_{prod} = (Q_{prod}, q_{0,\mathcal{P}}, \Sigma, \delta_{prod}, F_{prod})$, a (weighted and labeled) product SG is a tuple $\mathcal{G}_{prod} = (S_{prod}, U_C, U_A, P_{prod}, Acc, W)$, where $S_{prod} = S \times Q_{prod}$ is a finite set of states, U_C (resp. U_A) is a finite set of control inputs (resp. attack signals), $P_{prod}((\hat{s}, \hat{q}^1, \cdots, \hat{q}^n) | (s, q^1, \cdots, q^n), u_C, u_A) = P(s' | s, u_C, u_A)$ if $((q^1, \cdots, q^n), \mathcal{L}(\hat{s}), (\hat{q}^1, \cdots, \hat{q}^n)) \in \delta_{prod}$, $Acc = S \times F_{prod}$, and W is a weight function assigning each transition a reward.*

In the following, when the context is clear, we use s_{prod} to denote the state of product SG $\mathcal{G}_{prod}$. The weight function of product SG $\mathcal{G}_{prod}$ is defined as follows:

$$W(s'_{prod} | s_{prod}, u_C, u_A) = \sum_{i=1}^{n} I_i \left(s_{prod}, s'_{prod}\right) r(\phi_i), \tag{6.2}$$

where the $I(\phi_i)$ is the indicator function defined as

$$I_i \left(s_{prod}, s'_{prod}\right) = \begin{cases} 1, & \text{if } s_{prod} \notin S \times Q^1 \cdots F^i \cdots \times Q^n \text{ and } s'_{prod} \in S \times Q^1 \cdots F^i \cdots \times Q^n, \\ 0, & \text{otherwise.} \end{cases}$$

Note that the accepting states F^i for each automaton A_i are absorbing states. Thus by definition (6.2), a path on product SG $\mathcal{G}_{prod}$ collects rewards by satisfying the specifications in Φ at the first time.

In the following, we characterize the expected rewards that the controller and adversary can achieve via satisfying and violating the scLTL specification, respectively. Let $V_C(s_{prod})$ and $V_A(s_{prod})$ be the expected reward that the controller and adversary can obtain due to satisfying and violating specifications in Φ when starting from state $s_{prod} \in S_{prod}$, respectively. We then characterize the expected rewards V_C and V_A as follows.

Lemma 6.1. *The expected rewards of the controller and adversary induced by policy μ and τ can be represented as*

$$V_C(s_{prod}) = \sum_{u_C \in U_C} \big[\mu(s_{prod}, u_C) \sum_{u_A \in U_A} \tau(s_{prod}, u_A) \cdot \sum_{s'_{prod}} P_{prod}(s'_{prod}|s_{prod}, u_C, u_A)(W(s'_{prod}|s_{prod}, u_C, u_A) + V_C(s'_{prod}))\big], \tag{6.3}$$

$$V_A(s_{prod}) = \sum_{u_C \in U_C} \big[\tilde{\mu}(s_{prod}, u_C) \sum_{u_A \in U_A} \tau(s_{prod}, u_A) \cdot \sum_{s'_{prod}} P_{prod}(s'_{prod}|s_{prod}, u_C, u_A)(-W(s'_{prod}|s_{prod}, u_C, u_A) + V_A(s'_{prod}))\big]. \tag{6.4}$$

Moreover, given a pair of policies μ and τ, the expected reward of the controller and adversary are the unique solutions to the linear equations above.

Proof. Given a control strategy μ and adversary strategy τ, the product SG $\mathcal{G}_{prod}$ reduces to a policy-induced Markov chain (MC), whose transition reward at each state s_{prod} can be represented as

$$\tilde{W}(s_{prod}) = \sum_{u_C \in U_C} \mu(s_{prod}, u_C) \sum_{u_A \in U_A} \tau(s_{prod}, u_A) \cdot \sum_{s'_{prod}} P_{prod}(s'_{prod}|s_{prod}, u_C, u_A) W(s'_{prod}|s_{prod}, u_C, u_A). \tag{6.5}$$

Then the expected rewards $V_C(s_{prod})$ and $V_A(s_{prod})$ are interpreted as the expected reward collected by the path starting from s_{prod} to the set of absorbing states in MC. In the following, we prove that Enq. (6.3) and (6.4) hold by showing the equivalence of maximizing expected rewards and stochastic shortest path problem (SSPP) on policy-induced MC. An SSPP aims to find a path such that expected cost incurred along the path is minimized. Viewing transition reward $\tilde{W}$ as a transition cost $-\tilde{W}$, we then have that maximizing the expected reward $V_C(s_{prod})$ obtained by the controller is equivalent to an SSPP on the policy-induced MC. Analogous equivalence holds for the adversary. According to the dynamic programming algorithm for stochastic shortest path problem on MC [101], we can represent the expected

reward for the controller as

$$V_C(s_{prod}) = \tilde{W}(s_{prod}) + \sum_{s'_{prod}} P_{prod}(s'_{prod}|s_{prod}, u_C, u_A)V_C(s'_{prod}). \tag{6.6}$$

Substituting Eqn. (6.5) into Eqn. (6.6), we have that the expected rewards represented by Eqn. (6.3) and (6.4) hold.

The uniqueness follows from the fact that the expected reward for both players can be computed by solving a set of linear equations on the MC induced by control and adversary policies μ and τ. □

Functions V_C and V_A model the utility functions of the controller and adversary, respectively. By Lemma 6.1, we have that the game formulated in this chapter is nonzero-sum since $\tilde{\mu} \neq \mu$. We remark that when the adversary has perfect observation over control policy μ, the game reduces to a zero-sum game. Based on Lemma 6.1, we can narrow down our search space of control policy μ to the set of proper policies defined as follows.

Definition 6.3. (Proper Policies): *A stationary control policy μ is proper if under μ, regardless of the policy chosen by the adversary, the set of states in product SG $\mathcal{G}_{prod}$ that correspond to an accepting state in at least one automaton can eventually be reached with positive probability.*

If the specified accepting state cannot be reached with positive probability under some control policy μ, then μ is said to be improper. We formally state the result using the following proposition.

Proposition 6.1. *If a proper control policy μ' is associated with the highest expected reward for the controller among all proper policies, then it provides the highest expected reward among all stationary policies.*

Proof. Let μ be the control policy that enables the controller receiving highest reward among all stationary policies. If μ is a proper policy, then the result clearly holds. Next, we focus on the scenario where μ is improper and show that by construction, we have a proper control policy μ' such that the expected rewards for the controller under policies μ and μ' are equal. Divide the set of states S_{prod} into two subsets S_1 and S_2. Let S_1 be the set of states that cannot reach the set of destination states (under control policy μ), while S_2 denotes the set of states that reach the set of destination states with positive probability (under control policy μ). By the assumption on the existence of a proper policy $\tilde{\mu}$, we let $\mu'(s_{prod}) = \tilde{\mu}(s_{prod})$ for all $s_{prod} \in S_1$, and let $\mu'(s_{prod}) = \mu(s_{prod})$ for all $s_{prod} \in S_2$. Since μ is an improper policy while $\tilde{\mu}$ is a proper policy, we have that the expected reward received by the controller by committing to control policy μ' is no less than committing to μ when the initial state is in S_1. Suppose the initial state is in S_2. In the following, we show that the controller committing

to policy μ' receives expected reward no less than by committing to improper policy μ. The claim follows from the fact that the expected reward of the controller is computed as

$$V_C(s_{prod}) = \sum_{u_C \in U_C} \big[\mu(s_{prod}, u_C) \sum_{u_A \in U_A} \tau(s_{prod}, u_A) \cdot \sum_{s'_{prod}} P_{prod}(s'_{prod}|s_{prod}, u_C, u_A)(W(s'_{prod}|s_{prod}, u_C, u_A) + V_C(s'_{prod}))\big]. \quad (6.7)$$

Given that $\mu'(s_{prod}) = \mu(s_{prod})$ for all $s_{prod} \in S_2$ and the facts that $V(s_{prod}) = 0$ and $W(s_{prod}|s'_{prod}, u_C, u_A) = 0$ for all $s_{prod} \in S_1$, we have that the expected reward obtained by committing to policy μ' is no less than that of policy μ when the initial state is in S_2. By hypothesis on μ, we have that the proper control policy μ' corresponds to the highest expected reward when the initial state is in S_2. Hence, we have a proper control policy μ' such that the controller receives expected reward no less than committing to improper policy μ. □

6.3.2 A Mixed Integer Non-linear Programming Formulation

In this subsection, we formulate a mixed integer non-linear program (MINLP) to compute a mixed strategy for the controller using the product SG $\mathcal{G}_{prod}$. By Proposition 6.1, we can make the following two claims. First, we can restrict the search space of control policy to the set of proper control policies. Second, solving Problem 6.1 is equivalent to computing a control policy such that the expected reward V_C over the initial states is maximized.

Let χ be the probability distribution over the set of initial states. Then our objective is to maximize the expected reward of the controller over the set of initial states, i.e.,

$$\max_{\mu, \tau, V_C, V_A} \chi^\top V_C.$$

Next, we construct the set of constraints. Since we consider stationary mixed control strategy μ defined on product SG $\mathcal{G}_{prod}$, we have

$$\mu(s_{prod}, u_C) \geq 0, \ \forall s_{prod} \in S_{prod}, u_C \in U_C, \quad (6.8)$$

$$\sum_{u_C \in U_C} \mu(s_{prod}, u_C) = 1, \ \forall s_{prod} \in S_{prod}. \quad (6.9)$$

Constraints (6.8) and (6.9) give the probability simplex defined for control policy μ at each state s_{prod}. Given the stationary control policy μ, the product SG is then reduced to an Markov decision process (MDP) from the adversary's perspective. Then by [26], we have that it is sufficient to only consider pure adversary strategies. Thus, we have the following

constraints for adversary's strategy

$$\tau(s_{prod}, u_A) \in \{0, 1\},\ \forall s_{prod} \in S_{prod}, u_A \in U_A, \tag{6.10}$$

$$\sum_{u_A \in U_A} \tau(s_{prod}, u_A) = 1,\ \forall s_{prod} \in S_{prod}, \tag{6.11}$$

In the following, we define the constraints on controller's and adversary's expected rewards using big M method [113]. We define state-action value functions for the controller as $B_C(s_{prod}, \mu, u_A)$, where $B_C(s_{prod}, \mu, u_A)$ models the controller's expected reward when starting from state s_{prod} by committing to strategy μ and adversary playing action u_A. Following Lemma 6.1, we can represent $B_C(s_{prod}, \mu, u_A)$ as

$$B_C(s_{prod}, \mu, u_A) = \sum_{u_C} \mu(s_{prod}, u_C) \cdot \left[\sum_{s'_{prod}} P_{prod}(s'_{prod}|s_{prod}, u_C, u_A)(W(s'_{prod}|s_{prod}, u_C, u_A) + V_C(s'_{prod})) \right]. \tag{6.12}$$

Then the controller's expected reward $V_C(s_{prod})$ can be upper bounded for all s_{prod} and u_A as

$$V_C(s_{prod}) \leq B_C(s_{prod}, \mu, u_A) + (1 - \tau(s_{prod}, u_A))Z, \tag{6.13}$$

where Z is a sufficiently large positive number. In particular, when $\tau(s_{prod}, u_A) = 1$, i.e., the adversary's best response is to play action u_A, the RHS of (6.13) equals to $B_C(s_{prod}, \mu, u_A)$, when $\tau(s_{prod}, u_A) = 0$, i.e., the adversary's best response is not u_A, the constraint (6.13) becomes $V_C(s_{prod}) \leq \infty$, and it holds automatically.

Similarly, let $B_A(s_{prod}, \tilde{\mu}, u_A)$ be the state-action value function for the adversary, which models the adversary's expected reward when it perceives controller's strategy $\tilde{\mu}$ and plays action u_A, starting from state s_{prod}. Then $B_A(s_{prod}, \tilde{\mu}, u_A)$ can be represented as

$$B_A(s_{prod}, \tilde{\mu}, u_A) = \sum_{u_C} \tilde{\mu}(s_{prod}, u_C) \cdot \left[\sum_{s'_{prod}} P_{prod}(s'_{prod}|s_{prod}, u_C, u_A)(-W(s'_{prod}|s_{prod}, u_C, u_A) + V_A(s'_{prod})) \right], \tag{6.14}$$

where $\tilde{\mu}$ is obtained by the anchoring bias model (6.1). The expected reward for the adversary $V_A(s_{prod})$ can then be upper and lower bounded for all s_{prod} and u_A as follows:

$$B_A(s_{prod}, \tilde{\mu}, u_A) \leq V_A(s_{prod}) \leq B_A(s_{prod}, \tilde{\mu}, u_A) + (1 - \tau(s_{prod}, u_A))Z, \tag{6.15}$$

where Z is a sufficiently large positive number. When $\tau(s_{prod}, u_A) = 1$, i.e., the adver-

sary's best response is to play action u_A, constraint (6.15) is identical to Lemma 6.1, when $\tau(s_{prod}, u_A) = 0$, i.e., the adversary's best response is not u_A, constraint (6.15) can always be satisfied by tuning Z.

We finally state the MINLP formulation as follows:

$$\begin{aligned} \max_{\mu,\tau,V_C,V_A} \quad & \chi^\top V_C \\ \text{s.t.} \quad & \text{Eqn. (6.1) (6.8) (6.9) (6.10) (6.11) (6.13) and (6.15)} \end{aligned} \tag{6.16}$$

6.3.3 Exact Algorithm for Problem 6.1

In this subsection, we present a method to solve Problem 6.1 which is based on model reference adaptive search (MRAS). The method is shown to be probabilistically complete, i.e., with probability one the algorithm converges to the optimal solution. The idea behind MRAS method is that it makes the parameterized distribution $f(\cdot, \alpha)$ converges to a reference distribution that is implicitly encoded in the algorithm.

MRAS method focuses on (potentially nonconvex) optimization problem that has a unique global optimal solution. However, the MINLP formulated in this chapter possibly has infinitely many optimal solutions, i.e., the control policies in SE are not unique. In the following, we first validate the MRAS method under our problem setting by characterizing the reference distribution. Then we give details of the exact solution approach.

Let $\boldsymbol{\mu}$ be the set of feasible control policies. The controller's expected reward associated with control strategy $\mu \in \boldsymbol{\mu}$ is denoted as $V(\mu)$. Let $\boldsymbol{\mu}^*$ be the set of control policies in SE, and V^* be the corresponding controller's expected reward. We first make the following assumption [114].

Assumption 6.1. *For any $\epsilon < V^*$, the set of control policies $\{\mu : V(\mu) \geq \epsilon\} \cap \boldsymbol{\mu}$ has strictly positive Lebesgue measure.*

Assumption 6.1 ensures that any neighborhood of the set optimal control policy $\boldsymbol{\mu}^*$ has a positive probability of being selected. The exact algorithm we propose to solve Problem 6.1 is presented in Algorithm 9.

Algorithm 9 works as follows. For each state s_{prod}, a parameterized distribution, denoted as $f(\mu^{s_{prod}}, \alpha_k^{s_{prod}})$, defines the distribution over the admissible randomized control policies $\mu^{s_{prod}}$ at state s_{prod}. Here, the parameterized distribution at each state s_{prod} is set as the Dirichlet distribution. Since we consider stationary control policies, then the probability of selecting randomized control policy μ is defined by the joint distribution $\mathbf{f}(\mu, \alpha_k) = \prod_{s_{prod}} f(\mu^{s_{prod}}, \alpha_k^{s_{prod}})$, where α_k is the vector comprises $\alpha_k^{s_{prod}}$ for all s_{prod}. Algorithm 9 first initializes the iteration counter k and the distribution function $\mathbf{f}(\cdot, \alpha_0)$. Also a strictly increasing function $Y : \mathbb{R} \mapsto \mathbb{R}^+$ is specified (e.g., exponential function). Algorithm 9 constructs a sequence of nondecreasing thresholds $\{\bar{\gamma}_k : k = 0, 1, \cdots\}$. In particular, the

Algorithm 9 Exact Algorithm for Problem 6.1.

1: **Initialization:** Specify $\rho \in (0, 1]$, a positive number $\varepsilon \geq 0$, a strictly increasing function $Y : \mathbb{R} \mapsto \mathbb{R}^+$, and an initial p.d.f. $\mathbf{f}(\cdot, \alpha_0) > 0$ over the set of randomized control policies, i.e., $\mathbf{f} : \boldsymbol{\mu} \mapsto [0, 1]$. Set the iteration counter $k \leftarrow 0$.
2: **repeat**
3: Calculate the $(1 - \rho)$-quantile

$$\gamma_{k+1} = \sup_{l} \{l : \mathbb{P}_{\alpha_k}(V(\mu) \geq l) \geq \rho\}.$$

4: **if** $k = 0$ **then** set $\bar{\gamma}_{k+1} = \gamma_{k+1}$.
5: **else if** $k \geq 1$ **then**
6: **if** $\gamma_{k+1} \geq \bar{\gamma}_k + \epsilon$ **then** set $\bar{\gamma}_{k+1} = \gamma_{k+1}$.
7: **else** set $\bar{\gamma}_{k+1} = \bar{\gamma}_k$.
8: **end if**
9: **end if**
10: Compute the parameter vector α_{k+1} as

$$\alpha_{k+1} = \underset{\alpha \in \boldsymbol{\alpha}}{\operatorname{argmax}} \mathbb{E}_{\alpha_k} \left[\frac{[Y(V(\mu))]^k}{\mathbf{f}(\mu, \alpha_k)} I_{\{V(\mu) \geq \bar{\gamma}_{k+1}\}} \cdot \ln \mathbf{f}(\mu, \alpha) \right] \tag{6.17}$$

11: **until** $V(\mu_k) = V(\mu_{k-1}) = \cdots = V(\mu_{k-d})$

threshold $\bar{\gamma}_k$ at iteration k is constructed by computing the $(1-\rho)$ quantile γ_{k+1} as shown in line 4. By [115], given a value ρ_k and p.d.f. $\mathbf{f}(\mu, \alpha_k)$, the quantile evaluation at each iteration k can be accomplished by solving the following optimization problem.

$$\begin{aligned} \min_{l} \quad & \mathbb{E}_{\alpha_k} \phi(V(\mu), l) \\ \text{s.t.} \quad & l \in [0, V^*] \end{aligned}$$

where

$$\phi(V(\mu), v) = \begin{cases} (1-\rho_k)(V(\mu) - l), \text{ if } l \leq V(\mu) \\ \rho_k(l - V(\mu)), \text{ if } l \geq V(\mu) \end{cases}.$$

The optimization problem is convex in l and thus can be solved efficiently. If γ_{k+1} increases at least ε (line 7 is visited) , $\bar{\gamma}_{k+1}$ is updated to be γ_{k+1}. Otherwise the quantile value is set as the one from previous iteration. Algorithm 9 finally updates the parameter as shown in line 11. Given that $f(\cdot, \alpha_k^{s_{prod}})$ is the p.d.f. of Dirichlet distribution, the parameter updating rule (6.17) can be rewritten as

$$\alpha_{k+1}^{s_{prod}} = \operatorname*{argmax}_{\alpha} \mathbb{E}_{\alpha_k} \left[\frac{[Y(V(\mu))]^k}{\mathbf{f}(\mu, \alpha_k)} I_{\{V(\mu) \geq \bar{\gamma}_{k+1}\}} \ln f(\mu^{s_{prod}}, \alpha^{s_{prod}}) \right], \ \forall s_{prod} \in S_{prod}$$

We observe that the parameter update amounts on solving a convex program, and hence can be solved efficiently.

In the following, we characterize the parameter update rule [114].

Lemma 6.2. *The parameter α_k computed as shown in Algorithm 9 minimizes the K-L divergence between reference distribution $g_k(\mu)$ and $\mathbf{f}(\cdot, \alpha_k)$ at each iteration, where the reference distribution $g_k(\mu)$ is defined inductively as*

$$\begin{aligned} g_1(\mu) &= \frac{I_{\{V(\mu) \geq \bar{\gamma}_{k+1}\}}}{\mathbb{E}_{\alpha_0} \left[\frac{I_{\{V(\mu) \geq \bar{\gamma}_{k+1}\}}}{\mathbf{f}(\mu, \alpha_0)} \right]} \\ g_{k+1}(\mu) &= \frac{Y(V(\mu)) I_{\{V(\mu) \geq \bar{\gamma}_{k+1}\}} g_k(\mu)}{\mathbb{E}_{g_k} \left[Y(V(\mu)) I_{V(\mu) \geq \bar{\gamma}_{k+1}} \right]}, \ k = 1, 2, \cdots \end{aligned}$$

Proof. See [114] for the proof. □

By Lemma 6.2, we have that a sequence of reference distributions $g_k(\mu)$ are encoded in Algorithm 9, and the reference distribution is in the form of:

$$g_k(\mu) = \frac{V(\mu) g_{k-1}(\mu)}{\int_{\boldsymbol{\mu}} V(\mu) g_{k-1}(\mu) \, \mathrm{d}\mu}. \tag{6.18}$$

We characterize the reference distribution as follows:

Lemma 6.3. *The controller's expected reward with respect to the reference distribution $g(\mu)$ is monotone non-decreasing. Moreover, the expected reward converges to the optimal reward V^* as $k \to \infty$.*

Proof. By definition of reference distribution (6.18), we have

$$\begin{aligned}
\mathbb{E}_{g_k}[V(\mu)] &= \int_{\boldsymbol{\mu}} V(\mu) g_k(\mu)\,\mathrm{d}\mu \\
&= \int_{\boldsymbol{\mu}} V(\mu) \frac{V(\mu) g_{k-1}(\mu)}{\int_{\boldsymbol{\mu}} V(\mu) g_{k-1}(\mu)\,\mathrm{d}\mu}\,\mathrm{d}\mu \\
&= \frac{\mathbb{E}_{g_{k-1}}[V(\mu)^2]}{\mathbb{E}_{g_{k-1}}[V(\mu)]} \\
&\geq \mathbb{E}_{g_{k-1}}[V(\mu)],
\end{aligned}$$

where the third equality holds by the definition of expectation, and the last inequality holds by the fact that $\mathbb{E}[v^2] = \mathbb{E}[v]^2 + \sigma^2$, where v is a random variable and σ^2 is the variance of v. Thus we have that the expectation of controller's reward with respect to the reference distribution is monotone non-decreasing.

Next we show the sequence converges to the optimal reward V^* by contradiction. By definition, we have the expected reward is upper bounded by the optimal reward V^*. Suppose the sequence converges to some value $V' < V^*$. By definition (6.18), we rewrite the reference distribution at kth iteration as

$$g_k(\mu) = \prod_{i=1}^{k} \frac{V(\mu)^k}{\mathbb{E}_{g_i}[V(\mu)]} g_1(\mu). \tag{6.19}$$

Let $\bar{\boldsymbol{\mu}} = \{\mu | V(\mu) > V'\}$ be the set of control policies that can achieve reward no worse than V'. Given that $\mathbb{E}_{g_i}[V(\mu)]$ is monotone non-decreasing with respect to i and converges to V', we have $V(\mu)/\mathbb{E}_{g_i}[V(\mu)] > 1$ for all $1 \leq i \leq k$ and $\mu \in \bar{\boldsymbol{\mu}}$. Thus we have

$$\lim_{k\to\infty} g_k(\mu) \to \infty,\ \forall \mu \in \bar{\boldsymbol{\mu}}.$$

Then we obtain contradiction due to the following

$$1 = \liminf_{k\to\infty} \int_{\boldsymbol{\mu}} g_k(\mu)\,\mathrm{d}\mu \geq \liminf_{k\to\infty} \int_{\bar{\boldsymbol{\mu}}} g_k(\mu)\,\mathrm{d}\mu \geq \int_{\bar{\boldsymbol{\mu}}} \liminf_{k\to\infty} g_k(\mu)\,\mathrm{d}\mu = \infty,$$

where the first equality holds by the definition of p.d.f., the first inequality holds since $\bar{\boldsymbol{\mu}} \subseteq \boldsymbol{\mu}$, the second inequality holds by Fatou's Lemma, and the last equality holds by Assumption

6.1. Therefore we have

$$\lim_{k\to\infty} \mathbb{E}_{g_k}[V(\mu)] = V^*.$$

□

Lemma 6.3 characterizes the controller's expected reward with respect to the reference distribution. In the following, we show that the support of the reference distribution converges to the set of optimal control policies.

Lemma 6.4. *The reference distribution converges to a distribution whose support is contained in the set of optimal control policies $\boldsymbol{\mu}^*$.*

Proof. We prove by contradiction. Suppose the sequence converges to some a distribution whose support is $\boldsymbol{\mu}^* \cup \boldsymbol{\mu}'$, where $\boldsymbol{\mu}' \subseteq \boldsymbol{\mu}$ is a subset of control policies containing some non-optimal control policies. Therefore, there exists some control policy $\mu' \in \boldsymbol{\mu}'$ such that $\lim_{k\to\infty} g_k(\mu') > 0$. Moreover, since μ' is non-optimal, we have $V(\mu') < V^*$. Then we have $\lim_{k\to\infty} \mathbb{E}_{g_k}[V(\mu)] < V^*$, which contradicts Lemma 6.3. Hence, we have that the probability density at non-optimal policies converges to zero as $k \to \infty$. Since $\int_{\boldsymbol{\mu}} g_k(\mu)\,\mathrm{d}\mu = 1$, we further have that the support of the reference probability distribution is the set of optimal control policies $\boldsymbol{\mu}^*$ as $k \to \infty$. □

Lemma 6.4 implies that when the set of optimal control policies is discrete and finite, the reference distribution converges to n-point degenerate distribution as $k \to \infty$, with a Dirac delta function spike at each μ^* in SE; When the set of optimal control policies is continuous and infinite, the reference distribution converges to a hypersurface such that all optimal control policies lie on it.

Lemma 6.2, Lemma 6.3, and Lemma 6.4 imply that Algorithm 9 attempts to shrink the distribution $f(\cdot, \alpha^{s_{prod}})$ to the neighbor of optimal control policies by iteratively updating parameter as (6.17). Before presenting the optimal control policy, we first give some preliminary results [114].

Lemma 6.5. *Represent the distribution $f(\mu^{s_{prod}}, \alpha^{s_{prod}})$ in the form of exponential family as $f(\mu^{s_{prod}}, \alpha^{s_{prod}}) = h(\mu^{s_{prod}}) \exp\{\beta(\alpha^{s_{prod}}) L^{s_{prod}}(\mu^{s_{prod}}) - a(\beta)\}$ for some functions $h(\cdot)$, $\beta(\cdot)$, $L^{s_{prod}}(\cdot)$, and $A(\cdot)$. Then we have*

- *the equality $\mathbb{E}_{\alpha_{k+1}}[L^{s_{prod}}(\mu)] = \mathbb{E}_{g_{k+1}}[L^{s_{prod}}(\mu)]$ holds for all $k = 0, 1, \cdots$ and $s_{prod} \in S_{prod}$.*
- *the equality $\lim_{k\to\infty} \mathbb{E}_{\alpha_{k+1}}[L^{s_{prod}}(\mu)] = L^s(\mu^{s_{prod}^*})$ holds.*

In the following, we give the optimal control policy.

Lemma 6.6. *When the marginal distribution $f(\cdot, \alpha_k^{s_{prod}})$ is set as Dirichlet distribution, we have that*

$$\exp\left\{\lim_{k\to\infty} \mathbb{E}_{\alpha_k^{s_{prod}}}[\ln(\mu^{s_{prod}})]\right\} = \mu^{s_{prod}^*},\ \forall s_{prod} \in S_{prod}.$$

Proof. Rewrite the p.d.f. of Dirichlet distribution into the form of exponential family. By Lemma 6.5, we have that the optimal control policy satisfies

$$\lim_{k\to\infty} \mathbb{E}_{\alpha_k^{s_{prod}}}[\ln(\mu^{s_{prod}})] = \ln \mu^{s_{prod}^*},\ \forall s_{prod} \in S_{prod}$$

Taking the exponential on both sides, then we have the optimal control policy can be obtained as

$$\exp\left\{\lim_{k\to\infty} \mathbb{E}_{\alpha_k^{s_{prod}}}[\ln(\mu^{s_{prod}})]\right\} = \mu^{s_{prod}^*},\ \forall s_{prod} \in S_{prod}.$$

□

6.3.4 Approximate Algorithm for Problem 6.1

Although Algorithm 9 is probabilistically complete, its convergence rate suffers from the problem scale. In this subsection, we present a heuristic based approximate algorithm that returns a possible sub-optimal solution.

Algorithm 10 Approximate solution for problem 6.1

1: Sample a set of control policies $\mu_1, \cdots, \mu_H$
2: Let $\mathcal{H} \leftarrow \{V(\mu_1), \cdots, V(\mu_H)\}$, $\mathcal{V} \leftarrow \emptyset$,
3: **for** $V(\mu_h) \in \mathcal{H}$ **do**
4: $k \leftarrow 0$
5: **repeat**
6: Solve MILP (6.20) to obtain expected reward V_C^k.
7: $k \leftarrow k + 1$
8: **until** $\chi^\top V_C^k - \chi^\top V_C^{k-1} \leq \epsilon$ or MILP (6.20) is infeasible.
9: **if** termination condition $\chi^\top V_C^k - \chi^\top V_C^{k-1} \leq \epsilon$ is satisfied **then**
10: $\mathcal{V} \leftarrow \mathcal{V} \cup \{\chi^\top V_C^k\}$
11: **else** termination condition 'MILP (6.20) is infeasible' is satisfied
12: $\mathcal{V} \leftarrow \mathcal{V} \cup \{\chi^\top V_C^{k-1}\}$
13: **end if**
14: **end for**
15: **if** $\mathcal{V} = \emptyset$ **then**
16: Return to step 2
17: **else**
18: $h^* \leftarrow \text{argmax}\{V(\mu_h) | h = 1, \cdots, H\}$
19: $\mu \leftarrow$ policy obtained from $V(\mu_{h^*})$
20: **return** μ
21: **end if**

The approximate algorithm is detailed in Algorithm 10. Algorithm 10 first initializes a set of control policies $\mu_1, \cdots, \mu_H$. The initialization can be achieved by randomly sampling the control policies in $\boldsymbol{\mu}$. Given each sampled control policy μ_h, the product SG $\mathcal{G}$ reduces to an MDP. Then by solving the optimal control problem from the perspective of adversary on the MDP [34, 28, 30, 26], we can obtain the best response τ_h from the adversary. Given control and adversary policies μ_h and τ_h, we can compute the controller's expected reward V_C associated with the control policy μ_h, denoted as $V(\mu_h)$.

For each initial controller's expected reward $V(\mu_h)$, Algorithm 10 then invokes a value iteration based module (line 5 - line 8) to iteratively search for controller's expected reward. The value iteration module first initializes the iteration index k. Then it reduces the MINLP to MILP, which is detailed in Eqn. (6.20), to solve for a higher expected reward.

$$\begin{aligned}
\max_{\mu,\tau,V_C,V_A} \quad & \chi^\top V_C && (6.20)\\
\text{s.t.} \quad & V_C(s_{prod}) \leq B_C^k(s_{prod},\mu,u_A) + (1-\tau(s_{prod},u_A)Z,\ \forall s_{prod},u_A \\
& B_A^k(s_{prod},\mu,u_A) \leq V_A(s_{prod}) \leq B_A^k(s_{prod},\mu,u_A) + (1-\tau(s_{prod},u_A))Z,\ \forall s_{prod},u_A \\
& \text{Eqn. (6.1) (6.8) (6.9) (6.10) (6.11)}
\end{aligned}$$

where

$$\begin{aligned}
B_C^k(s_{prod},\mu,u_A) &= \sum_{u_C} \mu(s_{prod},u_C) \\
&\big[\sum_{s'_{prod}} P_{prod}(s'_{prod}|s_{prod},u_C,u_A)(W(s'_{prod}|s_{prod},u_C,u_A) + V_C^k(s'_{prod}))\big], && (6.21)\\
B_A^k(s_{prod},\tilde{\mu},u_A) &= \sum_{u_C} \tilde{\mu}(s_{prod},u_C) \\
&\big[\sum_{s'_{prod}} P_{prod}(s'_{prod}|s_{prod},u_C,u_A)(-W(s'_{prod}|s_{prod},u_C,u_A) + V_A^k(s'_{prod}))\big], && (6.22)
\end{aligned}$$

The nonlinear terms $B_C(s_{prod},\mu,u_A)$ and $B_A(s_{prod},\tilde{\mu},u_A)$ that appear in MINLP (6.16) are replaced by linear terms $B_C^k(s_{prod},\mu,u_A)$ and $B_A^k(s_{prod},\tilde{\mu},u_A)$. Thus the MINLP is converted to an MILP. The value iteration module terminates when the improvement between two iterations is below some threshold ϵ or the MILP (6.20) becomes infeasible. Then the set $\mathcal{V}$ is updated by adding the last feasible expected reward into it.

After H iterations, if no feasible expected reward is obtained, Algorithm 10 restarts from line 2. Otherwise, the control policy that gives the highest expected reward among $\mathcal{V}$ is returned as the result.

The convergence of Algorithm 10 is presented in the following theorem.

Theorem 6.1. *Algorithm 10 converges in finite time.*

Before presenting the proof of Theorem 6.1, we first introduce two operators denoted as

$T_\mu : \mathbb{R}^{|S_{prod}|} \to \mathbb{R}^{|S_{prod}|}$ and $T : \mathbb{R}^{|S_{prod}|} \to \mathbb{R}^{|S_{prod}|}$ as follows:

$$T_\mu V_C(s_{prod}) = \min_{\tau \in \mathcal{BR}(\tilde{\mu})} \sum_{u_C \in U_C} \mu(s_{prod}, u_C) \sum_{u_A \in U_A} \tau(s_{prod}, u_A) \sum_{s'_{prod}} \cdot \left[P_{prod}(s'_{prod} | s_{prod}, u_C, u_A)(W(s'_{prod} | s_{prod}, u_C, u_A) + V_C(s'_{prod})) \right], \tag{6.23}$$

$$TV_C(s_{prod}) = \max_{\mu} \min_{\tau \in \mathcal{BR}(\tilde{\mu})} \sum_{u_C \in U_C} \mu(s_{prod}, u_C) \sum_{u_A \in U_A} \tau(s_{prod}, u_A) \sum_{s'_{prod}} \cdot \left[P_{prod}(s'_{prod} | s_{prod}, u_C, u_A)(W(s'_{prod} | s_{prod}, u_C, u_A) + V_C(s'_{prod})) \right], \tag{6.24}$$

The following lemmas characterizes the operator T_μ.

Lemma 6.7. *For any vectors V and V' such that $V \leq V'$, we have $T_\mu^k V \leq T_\mu^k V'$ for all policies μ and k, where $T_\mu^k(\cdot)$ iteratively applying T_μ operator k times.*

Proof. By the definitions given in Eqn. (6.23) and (6.24), we can view the operator T_μ^k as the total expected reward collected from a k-stage problem with cost per stage $\tilde{W}^k(s_{prod})$. Increasing V is equivalent to increasing the terminal reward (e.g., the reward collected when reaching the destination) in the k-stage problem. Since cost per stage is fixed, hence increasing V will increase the expected total reward in the k-stage problem, which implies monotonicity of T_μ^k. □

Lemma 6.8. *Denote the expected reward induced by proper control policy μ and adversary policy $\tau \in \mathcal{BR}(\tilde{\mu})$ as $V_C^{\mu\tau}$. Then $V_C^{\mu\tau}$ satisfies $\lim_{M\to\infty}(T_\mu^M V_C) = V_C^{\mu\tau}$.*

Proof. Since we focus on stationary policies, then by inducting Lemma 6.1, $T_\mu^M V_C$ can be represented as

$$T_\mu^M V_C = {P_{prod}^{\mu\tau}}^M V_C + \sum_{m=0}^{M-1} \left(P_{prod}^{\mu\tau} \right)^m \tilde{W}, \tag{6.25}$$

where $P_{prod}^{\mu\tau}$ is the transition matrix of the Markov chain induced by control policy μ and adversary policy τ. Since the control policy μ is proper, we can eventually reach the set of destination states with probability 1. By definition (6.2), no reward can be collected when starting from destination states. Therefore, we have $\lim_{M\to\infty} P^{\mu\tau M} V_C = 0$. Then, by taking limit on both sides of Eqn. (6.25) as M tends to infinity, we have $\lim_{m\to\infty} T_\mu^M V_C = \lim_{M\to\infty} \sum_{m=0}^{M-1} \left(P_{prod}^{\mu\tau} \right)^m \tilde{W}$. By the definition of $V_C^{\mu\tau}$, we have $\lim_{M\to\infty}(T_\mu^M V_C) = V_C^{\mu\tau}$, and hence Lemma 6.8 is proved. □

Finally, we have the following proposition.

Proposition 6.2. *The optimal expected total reward for the controller at each iteration k satisfies $V_C^k = TV_C^{k-1}$.*

Proof. Suppose the expected reward for the controller is $\bar{V}_C^k$ at some iteration k such that $\bar{V}_C^k \neq TV_C^{k-1}$. If $\bar{V}_C^k > TV_C^{k-1}$, we have that $\bar{V}_C^k$ is not a feasible solution to MILP (6.20). If $\bar{V}_C^k < TV_C^{k-1}$, then starting from $\bar{V}_C^k$, we can always search along some direction in the feasible region of (6.20) until we reach the boundary of the feasible region to find some $\hat{V}_C^k \geq \bar{V}_C^k$. Hence, $\bar{V}_C^k$ is not the optimal solution to (6.20). Therefore, we have $V_C^k = TV_C^{k-1}$ holds. □

In the following, we present the proof of Theorem 6.1.

Proof. (Proof of Theorem 6.1.) We show that Algorithm 10 terminates within finite iterations because both outer and inner loops terminate within finite iterations.

First, the outer loop executes exactly $|\mathcal{H}|$ times and thus the outer loop terminates within finite iterations.

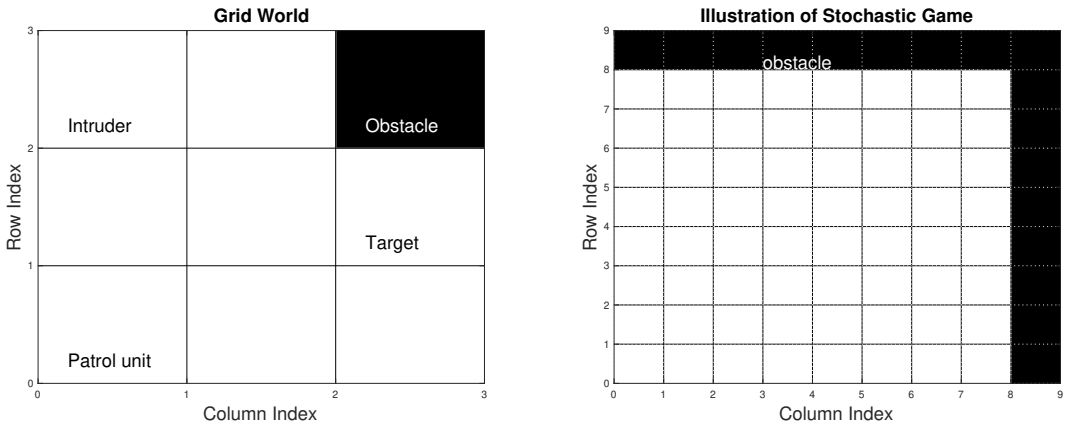
Next, we show at each outer loop iteration t, the value iteration module converges within finite time. It is obvious that the inner loop terminates when the initial guess on V_C is not feasible. In the following we focus on the feasible case. Let k be the iteration index of value iteration (line 3 to line 7). Let us denote the expected reward of the controller induced by control policy μ^k and adversary policy $\tau^k \in \mathcal{BR}(\tilde{\mu^k})$ at each iteration k as V_C^k. Let the expected reward of each transition starting from state s_{prod} and the transition matrix under control policy μ^k and adversary policy τ^k respectively be

$$\tilde{W}^k(s_{prod}) = \sum_{u_C} \sum_{u_A} \sum_{s'_{prod}} \mu^k(s_{prod}, u_C)\tau^k(s_{prod}, u_A)W(s'_{prod}|s_{prod}, u_C, u_A)$$

$$P^k(s'_{prod}|s_{prod}) = \sum_{u_C} \mu^k(s_{prod}, u_C) \sum_{u_A} \tau^k(s_{prod}, u_A)P_{prod}(s'_{prod}|s_{prod}, u_C, u_A)$$

By Lemma 6.1 and Proposition 6.2, we observe that $V_C^{k+1} = TV_C^k$ is equivalent to finding a control policy μ^{k+1} such that $T_{\mu^{k+1}}V_C^k = TV_C^k$. Therefore $V_C^k = T_{\mu^k}V_C^k = \tilde{W}^k + P^kV_C^k \leq \tilde{W}^{k+1} + P^{k+1}V_C^k = T_{\mu^{k+1}}V_C^k$, where the inequality holds by definition (6.23) and (6.24), i.e., $T_{\mu^k}V_C^k \leq TV_C^k$. View V_C^k as $T^0_{\mu^{k+1}}V_C^k$. Then composing $T_{\mu^{k+1}}$ m times and taking the limit as $m \to \infty$, by Lemma 6.7, we can construct a sequence of inequalities $V_C^k \leq T_{\mu^{k+1}}V_C^k, T_{\mu^{k+1}}V_C^k \leq T^2_{\mu^{k+1}}V_C^k, \cdots, T^{m-1}_{\mu^{k+1}}V_C^k \leq T^m_{\mu^{k+1}}V_C^k$. Therefore, we have $V_C^k \leq \lim_{m\to\infty} T^m_{\mu^{k+1}}V_C^k = V_C^{k+1}$, where the convergence of T_μ^m follows from Lemma 6.8. Hence, the expected reward increases with respect to the number of iterations k. Since V_C is upper bounded by $\sum_{\phi\in\Phi} r(\phi)$, we claim that the value iteration module converges within finite time. □

We finally remark that when the anchoring bias parameter $\lambda = 0$, Algorithm 10 becomes an exact algorithm, i.e., the control policy returned by Algorithm 10 is optimal. We also discuss the advantages and disadvantages of each algorithm. Algorithm 9 returns the optimal control policy with probability one. However, the convergence rate is relatively slow, and



(a) (b)

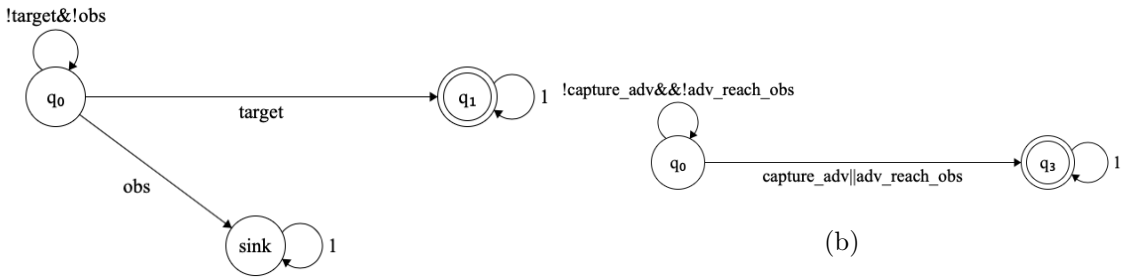
Figure 6-1: Fig. 6-1a presents the grid world that the patrol unit and intruder interact within. The initial locations of the patrol unit and intruder are $s_{1,1}$ and $s_{3,2}$, respectively. The target grid is $s_{2,3}$, and the obstacle region is $s_{3,3}$. Fig. 6-1b shows the illustration of the stochastic game. The row index gives the grid that the patrol unit is located in, while the column index gives the grid that the intruder is located in. The cells colored in black with row index 9 are the obstacles that the patrol unit needs to avoid, and the cells colored in black with column index 9 are the obstacles that the intruder needs to avoid. The cells located on the minor diagonal are the ones that models the event that the patrol unit captures the intruder, i.e., the locations of the patrol unit and intruder coincide.

hence the scalability of Algorithm 9 needs to be improved. Algorithm 10 addresses the slow convergence rate incurred by Algorithm 9. Moreover, it significantly reduces the memory and computation cost comparing to discretizaiton-based approach [113] and some other global optimization techniques such as spatial-and-bound [116]. However, Algorithm 10 does not provide completeness guarantee and it may return a sub-optimal control policy.

6.4 Case Study

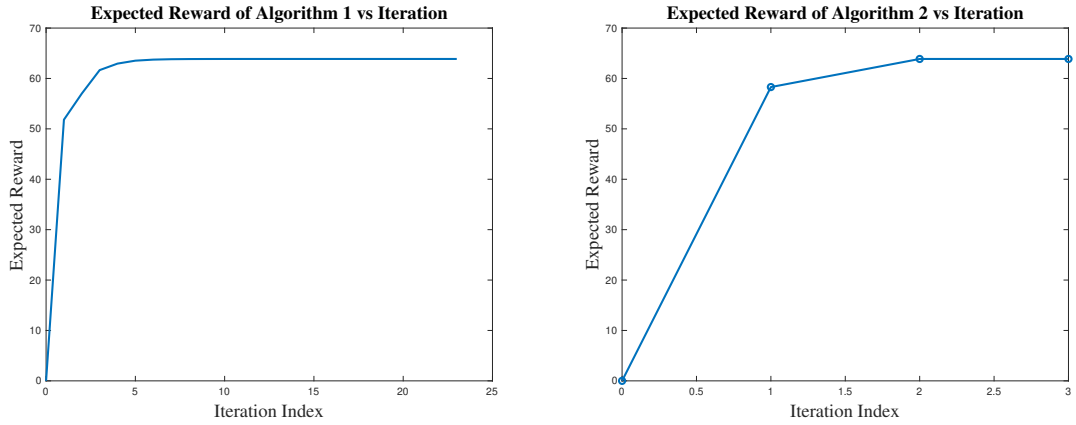
In this section, we present a numerical case study on patrolling security game involving a defender and adversary. The defender has one patrol unit, and the adversary has one intruder. The case study is implemented on a Macbook Pro with a 2.6GHz Intel Core i5 CPU and 8GB RAM.

We consider a PSG played in a 3×3 grid world with uniform grid size as shown in Fig. 6-1a. We index each grid s in the grid world using their row and column indices as $s_{r,c}$, where r is the row index and c is the column index. For instance, the grid located at bottom left corner is denoted as $s_{1,1}$, and the cell located at the upper right corner is denoted as



(a) (b)

Figure 6-2: The complete DFAs associated with specification ϕ_1 and ϕ_3. Fig.6-2a is the DFA for specification $\phi_1 = \neg obstacle$ U $target$. Fig. 6-2b is the DFA for specification $\phi_2 = \diamond capture\_intruder \vee \diamond adversary\_avoid\_obstacle$.



(a) (b)

Figure 6-3: Fig. 6-3a shows the expected reward that the patrol unit can obtain by committing to the control policy synthesized following Algorithm 9 at each iteration when the initial locations of the patrol unit and intruder are $s_{1,1}$ and $s_{3,1}$, respectively. Fig. 6-3b shows the expected reward for the patrol unit by committing to the control policy synthesized following Algorithm 10 at each iteration when the initial locations of the patrol unit and intruder are $s_{1,1}$ and $s_{3,1}$, respectively.

$s_{3,3}$. In the grid world, we label grid $s_{2,3}$ as the target region, and label grid $s_{3,3}$ as the obstacle region.

We consider a single patrol unit. At each grid, the actions available to the patrol unit are given as $U_C = \{N, S, W, E, stay\}$, where N, S, W, E represent moving towards north, south, west, and east, respectively. Action $stay$ indicates the patrol unit intends to stay at the current location.

There exists an intelligent adversary that can intrude the grids in the grid world. Due to the physical constraints, the adversary can not intrude arbitrary grid. In this case study, we assume that when the intruder is at grid $s_{r,c}$ at time t, it can only intrude the neighboring grids of $s_{r,c}$ at time $t+1$. For instance, if the intruder is located at $s_{3,1}$ at time t, it can only intruder one of the grids in $\{s_{3,2}, s_{2,1}\}$ at time $t+1$. To capture which state is to be intruded at the next time step, the action set of the intruder is given as $U_A = \{intrude\_N, intrude\_S, intrude\_E, intrude\_W\}$. The intruder does not have an action *stay* as the patrol unit does, and thus the intruder cannot intrude the same grid in two consecutive time steps. We assume the adversary has deterministic dynamics, that is, when the adversary issues command indicating the intrusion at grid $s_{r,c}$, no other grids will be intruded except $s_{r,c}$.

In this case study, the defender is given the following specifications. First, the vehicle is given a reach-avoid specification. Moreover, the patrol unit must capture the intruder or force the intruder collide with the obstacle. The specifications given to the defender are detailed as follows.

- The patrol unit must satisfy the reach-avoid specification, i.e., the patrol unit needs to eventually reach the target region while avoiding the obstacles. The specification is represented as $\phi_1 = \neg obstacle \text{ U } target$. The vehicle collects reward $r(\phi_1) = 20$ by satisfying specification ϕ_1.

- The vehicle needs to capture the intruder or force the intruder collide with the obstacle, i.e., $\phi_2 = \diamond capture\_intruder \vee \diamond adversary\_reach\_obstacle$. The vehicle collects reward $r(\phi_2) = 50$ by satisfying specification ϕ_2.

The complete automaton expressing the specifications are presented in Fig. 6-2a and Fig. 6-2b, respectively.

The patrol unit controlled by the defender and the intruder can each occupy one grid at each time, respectively. Thus to capture the joint locations of the patrol unit and the intruder, a stochastic game with 81 states is used as shown in Fig. 6-1b, i.e., the state set of SG is $S_g \times S_g$, where S_g is the set of grids as shown in Fig. 6-1a. For example, the state whose row and column indicies are 1 and 4 represents the scenario where the patrol unit is at grid 1 and the intruder is located at grid 4 in the grid world. The labeling function of the SG is given as follows. The cells colored in black with row index 9 and column index 9 in Fig. 6-1b represent the scenarios where the patrol unit and intruder collides with the obstacle in Fig. 6-1a, respectively. The target region is the set of states corresponding to row index 6 in Fig. 6-1b. The set of states lie on the diagonal of Fig. 6-1b are labelled to *capture_intruder*, i.e., the patrol unit and the intruder are located in the same grid in the grid world. The transition probability is as follow: the game has 0.8 probability to transit to the state intended by the patrol unit and $0.2/n$ to the neighbor states, where n is the number of neighbor states. For example, when the patrol unit is located at $s_{1,1}$ which is not

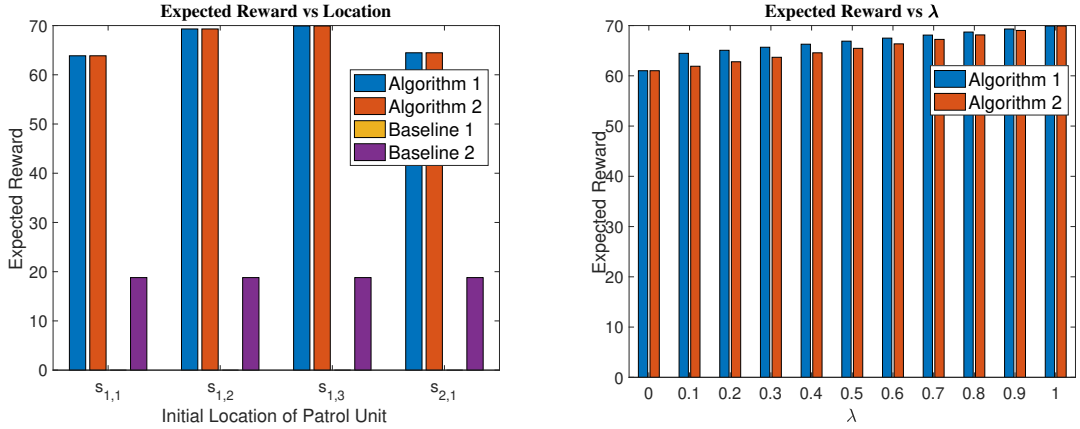
intruded by the intruder, we have

$$P((s_{1,2}, s_{3,2})|(s_{1,1}, s_{3,1}), E, intrude\_E) = 0.8,$$
$$P((s_{1,2}, s_{3,2})|(s_{1,1}, s_{3,1}), E, intrude\_E) = 0.1,$$
$$P((s_{1,2}, s_{3,2})|(s_{1,1}, s_{3,1}), E, intrude\_E) = 0.1,$$

In the following, we compare the proposed approach with two baselines. The first baseline, denoted as baseline 1 in the following, is a greedy approach, using which the patrol unit first attempts to capture the intruder or force the intruder running into the obstacle, and then navigates to the target region. Baseline 1 greedily minimizes the distance between the patrol unit and the intruder, and thus steers the patrol unit towards the current location of the intruder. In baseline 2, the patrol unit attempts to satisfy specification ϕ_1 first, and then it aims at satisfying specification ϕ_2. In both scenarios, the patrol unit commit to deterministic control policies. The performances of the two baselines are reported in Fig. 6-4a. We observe that the expected reward of baseline 1 is zero, indicating no specification is satisfied. The reason is that the intruder can never be captured by the patrol unit. Moreover, the intruder can always roam around the patrol unit to distract it from achieving the target region. In baseline 2, the patrol unit can reach the target region and hence collect the reward associated with specification ϕ_1. However, it still cannot capture the intruder or force it colliding with the obstacle.

We next give the simulation results using the Algorithm 9. The parameters are set as follows. We set the anchoring bias parameter $\lambda = 0$, i.e., the adversary has perfect observation over the control policy, and hence the worst-case scenario is considered in this case study. We let $\rho = 0.6$, $d = 3$, and $\varepsilon = 5$. Function $Y(\cdot)$ is set as the exponential function. Leveraging the p.d.f. of Dirichilet distribution, the parameter update (6.17) can be solved by numerically estimating the digamma function $\Gamma'(\mu)/\Gamma(\mu)$ for each state. We investigate the expected reward that the patrol unit can obtain when its initial location is $s_{1,1}$ and the initial location of the intruder is $s_{3,1}$. The relationship between the expected reward collected by the defender and iteration index is presented in Fig. 6-3a. The following observations are drawn from Fig. 6-3a. First, the expected reward of the defender is 63.9 given the synthesized policy. Second, the expected reward collected by the defender is monotone non-decreasing with respect to iteration index, and Algorithm 9 converges within 24 iterations. The comparison of the performances of Algorithm 9 and the two baselines are given in Fig. 6-4a. We have that Algorithm 9 achieves highest expected reward, while baseline 1 obtains 0 expected reward and baseline 2 gives expected reward that is higher than baseline 1 while smaller than the two proposed approaches.

We then report the performance of Algorithm 10. The expected reward that can be collected by the patrol unit using the control policy synthesized by Algorithm 10 is presented in Fig. 6-3b. We make the following observations. First, the heuristic solution approach gives expected reward 63.9, which is the same as Algorithm 9. The reason that Algorithm



(a) (b)

Figure 6-4: Fig. 6-4a compares the expected reward collected by the patrol unit using different approaches. Algorithm 9 achieves the highest expected reward. Algorithm 10 gives a sub-optimal solution. Baseline 1 and 2 achieves less expected reward compared with the proposed approaches. Fig. 6-4b gives the the expected reward collected by the patrol unit using Algorithm 9 and Algorithm 10 under different λ used in anchoring bias model (6.1). The value of λ varies from 0 to 1 with step size 0.1.

10 returns the optimal solution is that $\lambda = 0$ and the game becomes zero-sum. Second, Algorithm 10 converges faster compared to Algorithm 9. In particular, Algorithm 10 converges within 4 iterations. The comparison of the performances of Algorithm 10 and the two baselines are given in Fig. 6-4a. We have that Algorithm 10 achieves expected reward smaller than Algorithm 10 due to sub-optimality, while outperforms baseline 1 and baseline 2.

Next, we give the performances of Algorithm 9 and Algorithm 10 under different λ used in anchoring bias model (6.1). We vary the value of λ from 0 to 1, and compute the expected reward that the patrol unit obtains using Algorithm 9 and Algorithm 10. The expected rewards of the patrol unit under different λ using Algorithm 9 and Algorithm 10 are presented in Fig. 6-4b. We make the following observations. First, the patrol unit obtains lowest expected reward when $\lambda = 0$, i.e., when the adversary has perfect observation and plays its best response to the control policy. Second, the expected reward collected by the patrol unit increases as λ increases. The reason is that the adversary now has limited observation over the control policy, and hence the adversary's policy $\tau \notin \mathcal{BR}(\mu)$. In this case, the patrol unit collects higher reward compared with the scenario where $\lambda = 0$. The last observation is that the patrol unit obtains higher reward using Algorithm 9 than using Algorithm 10 when $\lambda \neq 0$. The reason is that Algorithm 10 may converge to a local optimal solution of MINLP (6.16) while Algorithm 9 returns optimal solution with probability one.

We finally give the computation time of both approaches. In this case study, there are 486 states in the product game. The construction of the product game takes 60 seconds. Algorithm 9 takes 592 seconds to converge, and Algorithm 10 takes 49 seconds to converge. This case study demonstrates the correctness and optimality of our approach. Improving the scalability by addressing the state explosion and optimizing the implementation in software is left to our future work.

6.5 Conclusion

In this chapter, we investigated minimum violation control synthesis on concurrent Stackelberg stochastic game. The players in the game are the controller and adversary. The controller is given a set of specifications modeled in scLTL, with each specification assigned a reward. The objective of the controller is to maximize the expected reward it can obtain by satisfying the given specifications. A linear anchoring bias model is adopted to model the limited observation capability of human adversaries. The minimum violation control synthesis is formulated as an MINLP on the product SG. We proposed two algorithms including an exact algorithm and an approximate algorithm. The exact algorithm is based on MRAS method. We validated the MRAS method under our problem setting. With probability one, the exact algorithm converges to an optimal control policy, regardless of the initial input of the algorithm. The approximate solution approach is a heuristic based algorithm and may converge to a sub-optimal control policy, depending on the initial input. A numerical case study on patrolling security game is presented to evaluate the proposed approaches. The formulation is limited to a fragment of LTL. Extensions to arbitrary LTL formulas are subject to future work.

Chapter 7

Optimal Secure Control under LTL Constraints in Partially Observable Environments

7.1 Introduction

Chapter 4 to 6 implicitly assume that the state of the stochastic game is fully observable by both the controller and the adversary. In practice, however, system states may not be observable. For example, as seen in [117], a robot might only have an estimate of its location based on the output of a vision sensor. The inability to observe all states necessitates the use of a framework that accounts for *partial observability*. For LTL formula satisfaction in partially observable environments with a single agent, partially-observable Markov decision processes (POMDPs) can be used to model and solve the problem [77, 1]. However, determining an 'optimal policy' for an agent in a partially observable environment is NP-hard for the infinite horizon case, which was shown in [31]. This demonstrates the need for techniques to determine approximate solutions.

In this chapter, we study the problem of determining strategies for an controller that has to satisfy an LTL formula in the presence of an adversary in a partially observable environment. The controller and the adversary follow a concurrent Stackelberg setting. Their actions jointly influece the transitions between states. The goal for the controller will be to synthesize a policy that will maximize the probability of satisfying an LTL formula for any adversary policy. The policies of the players are represented as finite state controllers (FSCs). We make the following specific contributions:

- We show that maximizing the satisfaction probability of the LTL formula under any adversary policy is equivalent to maximizing the probability of reaching a recurrent set of a Markov chain constructed by composing representations of the environment, the LTL objective, and the respective agents' controllers.

- We develop a heuristic algorithm to determine controller and adversary FSCs of fixed sizes that will satisfy the LTL formula with nonzero probability, and show that it is sound. The search for a controller's policy that will maximize the probability of satisfaction of the LTL formula for any adversary policy can then be reduced to a search among these FSCs.

- We propose a procedure based on value-iteration that maximizes the probability of satisfying the LTL formula under fixed controller and adversary FSCs. This satisfaction probability is related to a Stackelberg equilibrium of a partially observable stochastic game involving the controller and adversary. We also give guarantees on the convergence of this procedure.

- We study the case when the size of the controller FSC can be changed to improve the satisfaction probability.

- We present an example to illustrate our approach.

The remainder of this chapter is organized as follows. Section 7.2 presents the concept of finite state controller and the problem formulation. We give a sound solution approach in Section 7.3. A numerical case study is presented in Section 7.4. We conclude this chapter in Section 7.5.

7.2 System Model and Problem Formulation

To capture the inability for the controller and adversary to perfectly observe the state, we introduce a concept named partially observable stochastic game (POSG), which extends SG introduced in Definition 3.3. In a POSG, each player receives an observation that is derived from the state. A POSG is defined as follows.

Definition 7.1 (Partially Observable Stochastic Game)**.** *A* partially observable stochastic game *is defined by the tuple* $\mathcal{G} := (S, S_0, U_C, U_A, P, \mathcal{O}_C, \mathcal{O}_A, O_C, O_A, \Pi, \mathcal{L})$*, where* $S, S_0, U_C, U_A, P, \Pi, \mathcal{L}$ *are as in Definition 3.3.* $\mathcal{O}_C$ *and* $\mathcal{O}_A$ *denote the (finite) sets of observations available to the controller and adversary.* $O_* : S \times \mathcal{O}_* \to [0, 1]$ *encodes* $\mathbb{P}(o_*|s)$*, where* $* \in \{C, A\}$.

The initial state of the system belongs to $S_0 \subseteq S$. A transition from a state s_t to the next state s_{t+1} at step t is determined jointly by the respective actions u_C and u_A of the controller and adversary according to the transition probability function P. The functions O_* model imperfect sensing. In order for O_* to satisfy the conditions of a probability distribution, we need $O_*(o|s) \geq 0,\ \forall o \in \mathcal{O}_*$ and $\sum_{o \in \mathcal{O}_*} O_*(o|s) = 1$, where $* \in \{C, A\}$.

The goal of the controller is to synthesize a policy μ, which will be detailed later, to maximize the probability of satisfying an LTL specification φ. The objective of the adversary

is to synthesize a policy τ so as to minimize the satisfaction probability of φ, given the controller's policy.

At a state s_t, the adversary makes an observation, O_A^t of the state according to O_A. The adversary is also assumed to be aware of the policy μ committed to by the controller. Therefore, the overall information available to the adversary is $\mathfrak{I}_A^t := \bigcup_{i=0:t} O_A^i \cup \{\mu\}$.

Different from the information available to the adversary, at state s_t, the controller makes an observation O_C^t of the state according to O_C. Therefore, the overall information for the controller is $\mathfrak{I}_C^t := \bigcup_{i=0:t} O_C^i$.

We now define the policies on POSG.

Definition 7.2 (Controller's Policy on POSG)**.** *A* controller's policy *defined on the POSG is a map from the respective overall information to a probability distribution over the corresponding action space, i.e.* $\mu^t : \mathfrak{I}_C^t \times U_C \to [0, 1]$.

Similarly, the adversary's policy τ^t is defined as $\tau^t : \mathfrak{I}_A^t \times U_A \to [0, 1]$. Note that the policies considered in this chapter are mixed strategies. In the sequel, we will use finite state controllers (FSCs) as a representation of policies. An FSC consist of a finite number of internal states. Transitions between the states of an FSC is governed by the current observation of the agent. In our setting, we will have two FSCs, one for the controller and another for the adversary. We will then limit the search for controller and adversary policies to one over FSCs of fixed cardinality.

Definition 7.3 (Finite State Controller)**.** *A* finite state controller for the controller*, denoted $\mathcal{C}_C$, is a tuple $\mathcal{C}_C = (G_C, g_{0,C}, \mu)$, where G_C is a finite set of (internal) states of the controller, $g_{0,C}$ is the initial state of the FSC, and $\mu : G_C \times \mathcal{O}_C \times G_C \times U_C \to [0, 1]$, written $\mu(g'_C, u_C | g_C, o_C)$, is a probability distribution of the next internal state and action, given a current internal state and observation.*

The size of an FSC $\mathcal{C}_C$ is given as $|G_C|$. A finite state controller $\mathcal{C}_A = (G_A, g_{0,A}, \tau)$ for the adversary can be defined in a similar manner. We also define the set of proper FSCs as follows.

Definition 7.4 (Proper FSCs)**.** *An FSC is* proper *with respect to an LTL formula φ if there is a positive probability of satisfying φ under this policy in an environment represented as a partially observable stochastic game $\mathcal{G}$.*

An FSC is a finite-state probabilistic automaton that takes the current observation of the agent as its input, and produces a distribution over the actions as its output. In this chapter, we assume that the size of the adversary FSC is fixed, and known to the controller. This can be interpreted as one way for the controller to have knowledge of the capabilities of an adversary. The problem investigated in this chapter is stated below.

Problem 7.1. *Given a POSG $\mathcal{G}$ and an LTL formula φ, determine a controller's policy specified by an FSC $\mathcal{C}_C$ that maximizes the probability of satisfying LTL formula φ under any adversary policy τ that is represented as an FSC $\mathcal{C}_A$ of fixed size.*

7.3 Control Synthesis to Maximize Satisfaction Probability with Partial Observability

In this section, we present the proposed solution approach to Problem 7.1. We first construct a product POSG. Then we compute recurrent sets on the POSG, and relate the probability of the LTL specification being satisfied by the product POSG to the probability of reaching the recurrent sets. We next determine the candidate finite state controllers to represent the policies. We finally develop a value iteration based algorithm to compute the controller's policy.

7.3.1 Product POSG and Recurrent Sets

In order to find runs on $\mathcal{G}$ that would be accepted by a DRA $\mathcal{R}$ built from an LTL formula φ, we construct a product POSG defined as follows.

Definition 7.5 (Product POSG)**.** *Given $\mathcal{G}$ and $\mathcal{R}$ (built from LTL formula φ), a* product POSG *is $\mathcal{G}_{prod} = (S_{prod}, S_{0,prod}, U_C, U_A, P_{prod}, \mathcal{O}_C, \mathcal{O}_A, O_{C,prod}, O_{A,prod}, F_{prod}, \Pi, \mathcal{L}_{prod})$, where*

- $S_{prod} = S \times Q$.
- $S_0^{\varphi} = \{(s_0, q_0) : s_0 \in S_0\}$.
- $O_{*,prod}(o|(s,q)) = O_*(o|s)$ *with* $* \in \{C, A\}$
- $P_{prod}((s',q')|(s,q), u_C, u_A) = P(s'|s, u_C, u_A)$ *iff* $\delta(q, \mathcal{L}(s')) = q'$, *and* 0 *otherwise.*
- $F_{prod} = \{(L_{prod}(i), K_{prod}(i))\}_{i=1}^{Z}$, $L_{prod}(i), K_{prod}(i) \subset S_{prod}$, *and* $(s,q) \in L_{prod}(i)$ *iff* $q \in L(i)$, $(s,q) \in K_{prod}(i)$ *iff* $q \in K(i)$.
- $\mathcal{L}_{prod}((s,q)) = \mathcal{L}(s)$.

From Definition 7.5, it is clear that acceptance conditions in the product POSG depend on the DRA while the transition probabilities of the product POSG are determined by transition probabilities of the original POSG. Therefore, a run on the product POSG can be used to generate a path on the POSG and a run on the DRA. Then, if the run on the DRA is accepting, we say that the product POSG satisfies the LTL specification φ.

The FSCs $\mathcal{C}_C$ and $\mathcal{C}_A$, when composed with $\mathcal{G}_{prod}$, will result in a finite-state, (fully observable) policy-induced Markov chain. To maintain consistency with the literature, we will refer to this as the *global Markov chain (GMC)* [77] defined as follows.

Definition 7.6 (Global Markov Chain (GMC))**.** *The* GMC *resulting from a product-POSG $\mathcal{G}_{prod}$ controlled by FSCs $\mathcal{C}_C$ and $\mathcal{C}_A$ is $\mathcal{M} := \mathcal{M}^{\varphi,\mathcal{C}_C,\mathcal{C}_A} = (\bar{S}, \bar{s}_0, \bar{P}, \Pi, \bar{\mathcal{L}})$, where*

- $\bar{S} = S_{prod} \times G_C \times G_A$.

- $\bar{S}_0 = \{(s_0, q_0, g_{0,C}, g_{0,A}) : s_0 \in S_0\}$.
- $\bar{P}$ *is given by*

$$\begin{aligned}\bar{P} :=& P^{\varphi,\mathcal{C}_C,\mathcal{C}_A}((s',q'),g'_C,g'_A|(s,q),g_C,g_A) \\ =& \sum_{o\in\mathcal{O}_C}\sum_{o'\in\mathcal{O}_A}\sum_{u_C}\sum_{u_A} O_C(o|s)O_A(o'|s)\mu(g'_C,u_C|g_C,o)\tau(g'_A,u_A|g_A,o') \\ & \cdot P_{prod}((s',q')|(s,q),u_C,u_A)\end{aligned} \tag{7.1}$$

- $\bar{\mathcal{L}} = \mathcal{L}_{prod}((s,q))$.

Similar to $\mathcal{G}_{prod}$, the Rabin acceptance condition for $\bar{\mathcal{M}}$ is: $\bar{F} = \{(\bar{L}(i), \bar{K}(i))\}_{i=1}^{Z}$, with $(s,q,g_C,g_A) \in \bar{L}(i)$ iff $(s,q) \in L_{prod}(i)$ and $(s,q,g_C,g_A) \in \bar{K}(i)$ iff $(s,q) \in K_{prod}(i)$. A state of $\mathcal{M}$ is $\mathfrak{s} := (s,q,g_C,g_A)$. A path on $\mathcal{M}$ is a sequence $\pi := \mathfrak{s}_0\mathfrak{s}_1\ldots$ such that $\bar{P}(\mathfrak{s}_{k+1}|\mathfrak{s}_k) > 0$, where $\bar{P}(\cdot)$ is the transition probability in $\mathcal{M}$. The path is accepting if it satisfies the Rabin acceptance condition. This corresponds to an execution in $\mathcal{G}_{prod}$ controlled by $\mathcal{C}_C$ and $\mathcal{C}_A$. To quantitatively reason about $\mathcal{M}$, we define a probability space following the treatment in [18]. The set of paths in $\mathcal{M}$, denoted $\texttt{Paths}(\mathcal{M})$ forms the sample space. The set of events $\mathcal{F}$ is the smallest $\sigma-$algebra generated by *cylinder sets* spanned by path fragments of finite length in $\mathcal{M}$. The cylinder set spanned by $\hat{\pi} := \mathfrak{s}_0\mathfrak{s}_1\ldots\mathfrak{s}_n$ is given by paths $\pi \in \texttt{Paths}(\mathcal{M})$ that start with $\hat{\pi}$. This is denoted $Cyl(\mathfrak{s}_0\mathfrak{s}_1\ldots\mathfrak{s}_n)$. Then, the (unique) probability measure on $\mathcal{F}$ for the events is given by $\mathbb{P}^{\mathcal{M}}(Cyl(\mathfrak{s}_0\mathfrak{s}_1\ldots\mathfrak{s}_n)) := \mathbb{P}(\mathfrak{s}_0)\bar{P}(\mathfrak{s}_1|\mathfrak{s}_0)\bar{P}(\mathfrak{s}_2|\mathfrak{s}_1)\ldots\bar{P}(\mathfrak{s}_n|\mathfrak{s}_{n-1})$. The probability of the LTL objective φ being satisfied in state $\mathfrak{s}$ is $\mathbb{P}^{\mathcal{M}}(s \models \varphi) := \mathbb{P}^{\mathcal{M}}\{\pi \in \texttt{Paths}(\mathfrak{s})|\pi \models \varphi\}$. In the sequel, we write $\mathbb{P}(\mathcal{M} \models \varphi) := \mathbb{P}(\mathcal{M}^{\varphi,\mathcal{C}_C,\mathcal{C}_A} \models \varphi)$ to denote $\mathbb{P}^{\mathcal{M}}(s_0 \models \varphi)$. We direct the reader to Section 10.1 in [18] for a characterization of the probability spaces for different LTL objectives.

Let $\mathcal{E} = \mathcal{E}^{\varphi,\mathcal{C}_C,\mathcal{C}_A}$ denote the recurrent states of $\mathcal{M}$ under FSCs $\mathcal{C}_C$ and $\mathcal{C}_A$. Let $E^S := (s,q)$ be the restriction of a recurrent state to a state of $\mathcal{G}_{prod}$.

Proposition 7.1. $\mathbb{P}(\mathcal{M} \models \varphi) = \mathbb{P}(\mathcal{G}_{prod} \models \varphi|\mathcal{C}_C,\mathcal{C}_A) > 0$ *if and only if there exists* $\mathcal{C}_C$ *such that for any* $\mathcal{C}_A$*, there exists a Rabin acceptance pair* $(L_{prod}(i), K_{prod}(i))$ *and an initial state of* $\mathcal{M}$*,* $\bar{s}_0$*, where the following conditions hold:*

$$K_{prod}(i) \cap \mathcal{E}^S \neq \emptyset \tag{7.2a}$$

$$(K_{prod}(i) \times G_C \times G_A) \cap \mathcal{R} \text{ is accessible from } \bar{s}_0 \tag{7.2b}$$

$$(L_{prod}(i) \times G_C \times G_A) \cap \mathcal{R} \text{ is not accessible from } \bar{s}_0 \tag{7.2c}$$

Proof. If for every $(L_{prod}(i), K_{prod}(i))$, at least one of the conditions in Eqn. (7.2) does not hold, then at least one of the following statements is true: *i)*: no state that has to be visited infinitely often is recurrent; *ii)*: there is no initial state from which a recurrent state that has to be visited infinitely often is accessible; *iii)*: some state that has to be visited only finitely often in steady state is recurrent. This means $\mathcal{G}_{prod} \not\models \varphi$ for all $\mathcal{C}_C$.

Conversely, if all the conditions in Eqn. (7.2) hold for some $(L_{prod}(i), K_{prod}(i))$, then $\mathcal{G}_{prod} \models \varphi$ by construction. □

To quantify the satisfaction probability for a controller's policy under any adversary's policy, assume that the recurrent states of $\mathcal{M}$ are partitioned into recurrence classes $\{E_1, \ldots, E_p\}$. This partition is maximal, in the sense that two recurrent classes cannot be combined to form a larger recurrent class, and all states within a given recurrent class communicate with each other [1].

Definition 7.7 ($\varphi-$feasible Recurrent Set)**.** *A recurrent set E_k is $\varphi-$*feasible *under FSCs $\mathcal{C}_C$ and $\mathcal{C}_A$ if there exists $(L_{prod}(i), K_{prod}(i))$ such that $K_{prod}(i) \cap E_k^S \neq \emptyset$ and $L_{prod}(i) \cap E_k^S = \emptyset$.*

Let $\varphi - RecSets^{\mathcal{C}_C, \mathcal{C}_A}$ denote the set of $\varphi-$feasible recurrent sets under the respective FSCs. Let $\pi \to \mathcal{E}$ be the event that a path of $\mathcal{M}$ will reach a recurrent set. Algorithm 11 returns $\varphi-$feasible recurrent sets of the GMC $\mathcal{M}$ constructed by $\mathcal{G}_{prod}$ under fixed FSCs $\mathcal{C}_C, \mathcal{C}_A$. Given the $\varphi-$feasible recurrent sets of the GMC $\mathcal{M}$, we can relate the satisfaction probability with the reachability probability as shown in the following theorem.

Theorem 7.1. *The probability of satisfying an LTL formula φ in a POSG with policies $\mathcal{C}_C$ and $\mathcal{C}_A$ is equal to the probability of paths in the GMC (under the same FSCs) reaching $\varphi-$feasible recurrent sets. That is,*

$$\mathbb{P}(\mathcal{G}_{prod} \models \varphi | \mathcal{C}_C, \mathcal{C}_A) = \sum_{E \in \varphi - RecSets^{\mathcal{C}_C, \mathcal{C}_A}} \mathbb{P}(\textit{reach } E) \tag{7.3}$$

Proof. Since the recurrence classes are maximal, $\mathbb{P}(\text{reach } (E_1 \cup \cdots \cup E_p)) = \sum_{k=1}^{p} \mathbb{P}(\text{reach } E_k)$. From Definition 7.7, a $\varphi-$feasible recurrent set will necessarily contain a Rabin acceptance pair. Therefore, the probability of $\mathcal{G}_{prod}$ satisfying the LTL formula under $\mathcal{C}_C$ and $\mathcal{C}_A$ is equivalent to the probability of paths on $\mathcal{M}$ leading to $\varphi-$feasible recurrent sets, which is given by Eqn. (7.3). □

Theorem 7.1 renders the following corollary.

Corollary 7.1. *From Theorem 7.1, it follows that:*

$$\max_{\mathcal{C}_C} \min_{\mathcal{C}_A} \mathbb{P}(\mathcal{M} \models \varphi) = \max_{\mathcal{C}_C} \min_{\mathcal{C}_A} \mathbb{P}(\mathcal{G}_{prod} \models \varphi | \mathcal{C}_C, \mathcal{C}_A) \tag{7.4a}$$

$$= \max_{\mathcal{C}_C} \min_{\mathcal{C}_A} \sum_{E \in \varphi - RecSets^{\mathcal{C}_C, \mathcal{C}_A}} \mathbb{P}(\textit{reach } E) \tag{7.4b}$$

We remark that Proposition 7.1, Theorem 7.1, and Corollary 7.1 address a broader class of problems than in Problem 7.1 since they do not assume that the size of the adversary FSC is fixed.

Algorithm 11 Generate $\varphi-$feasible Recurrent Sets for $\mathcal{G}_{prod}$ under FSCs $\mathcal{C}_C, \mathcal{C}_A$

Input: $\mathcal{M} := \mathcal{M}^{\varphi,\mathcal{C}_C,\mathcal{C}_A}, \{L_{prod}(i), K_{prod}(i)\}_{i=1}^{Z}$
Output: $\{E_k\}$, that is recurrent and $\varphi-$feasible
Induce digraph $\mathcal{DG}$ of $\mathcal{M}$ as $(\mathfrak{S}, \mathcal{D})$, s.t. $\forall \mathfrak{s}_1, \mathfrak{s}_2 \in \mathfrak{S} : \mathfrak{s}_1 \to \mathfrak{s}_2 \in \mathcal{D} \Leftrightarrow \bar{P}(\mathfrak{s}_2|\mathfrak{s}_1) > 0$.
$\mathcal{C} = SCCs(\mathcal{DG}) = \{C_1, \ldots, C_N\}$ ▷ *strongly connected components of digraph*
$RecSets := \{E_1, \ldots, E_p\}$ such that $E_i \in \mathcal{C}$ and E_i is a *sink SCC*
$\varphi - RecSets^{\mathcal{C}_C,\mathcal{C}_A} = \emptyset$
for $j = 1$ to p **do**
    for $i = 1$ to M **do**
        if $(L_{prod}(i) \cap E_j^S = \emptyset) \wedge (K_{prod}(i) \cap E_j^S \neq \emptyset)$ **then**
            $\varphi - RecSets^{\mathcal{C}_C,\mathcal{C}_A} = \varphi - RecSets^{\mathcal{C}_C,\mathcal{C}_A} \cup E_j$
        end if
    end for
end for

7.3.2 Determining Candidate FSCs of Fixed Sizes

In this subsection, we present a heuristic procedure that uses only the most recent observations of the controller and adversary to generate a set of admissible FSC structures such that the resulting GMC will have a $\varphi-$feasible recurrent set. We show that the procedure has a computational complexity that is polynomial in the number of states of the GMC and additionally establish that this algorithm is sound.

Let $\mathcal{I}_C : G_C \times \mathcal{O}_C \times G_C \times U_C \to \{0, 1\}$, where $\mathcal{I}_C(g', u_C|g, o) = 1 \Leftrightarrow \mu_C(g', u_C|g, o_C) > 0$. $\mathcal{I}_C(\cdot)$ shows if an observation o_C can enable the transition from an FSC state g to g' while issuing action u_C. We also assume that $\forall (g, o_C) \in G_C \times \mathcal{O}_C, \exists (g', u_C) \in G_C \times U_C$ such that $\mathcal{I}_C(g', u_C|g, o_C) = 1$ [1]. We define $\mathcal{I}_A : G_A \times \mathcal{O}_A \times G_A \times U_A \to \{0, 1\}$ in an analogous manner.

In Algorithm 12, for controller and adversary FSCs with fixed number of states, we determine candidate $\mathcal{C}_C$ and $\mathcal{C}_A$ such that the resulting $\mathcal{M}$ will have a $\varphi-$feasible recurrent set. We start with initial candidate structures $\mathcal{I}_C^o$ and $\mathcal{I}_A^o$ and induce the digraph of the resulting GMC (*Line 1*). In our case, $\mathcal{I}_*^o$ is such that $\mathcal{I}_*^o(g'_*, u_*|g_*, o_*) = 1$ for all g'_*, g_*, u_*, o_*. We first determine the set of communicating classes of the GMC, which is equivalent to determining the strongly connected components (SCCs) of the induced digraph (*Line 3*). A communicating class will be recurrent if it is a *sink* SCC of the corresponding digraph. The states in Bad_i are those in C that are part of the Rabin accepting pair that has to be visited only finitely many times (and therefore, to be visited with very low probability in steady state) (*Line 6*). Bad_i further contains states that can be transitioned to from some state in C. This is because once the system transitions out of C, it will not be able to return to it in order to satisfy the Rabin acceptance condition (*Line 5*) (and hence, C will not be recurrent). $Good_i$ contains those states in C that need to be visited infinitely often according to the Rabin acceptance condition (*Line 7*).

$$O_C(o_C|s)O_A(o_A|s)\mu(g'_C, u_C|g_C, o_C)\tau(g'_A, u_A|g_A, o_A)P_{prod}((s', q')|(s, q), u_C, u_A) > 0 \quad (7.5)$$

$$O_C(o_C|s)O_A(o_A|s)\mu(g''_C, u_C|g_C, o_C)\tau(g''_A, u_A|g_A, o_A)P_{prod}((s'', q'')|(s, q), u_C, u_A) > 0 \quad (7.6)$$

The agents have access to a state only via their observations. A controller action is forbidden if there exists an adversary action that will allow a transition to a state in Bad_i under observations o_C and o_A. This is achieved by setting corresponding entries in $\mathcal{I}_C$ to zero (*Lines 12-17*). An adversary action is not useful if for every controller action, the probability of transitioning to a state in $Good_i$ is nonzero under o_C and o_A. This is achieved by setting the corresponding entry in $\mathcal{I}_A$ to zero (*Lines 18-23*). The complexity of Algorithm 12 is given as follows.

Proposition 7.2. *Define* $|\mathcal{O}| = |\mathcal{O}_C| + |\mathcal{O}_A|$ *and* $|U| = |U_C| + |U_A|$. *Then, Algorithm 12 has an overall computational complexity of* $\mathbf{O}(|S|^2|G_C|^2|G_A|^2|\mathcal{O}||U|)$.

Proof. The overall complexity depends on: *(i)* Determining strongly connected components (*Line 3*): This can be done in $\mathbf{O}(|\mathfrak{S}| + |\mathcal{D}|)$ [118]. Since $|\mathfrak{S}| = |S||G_C||G_A|$ and $|\mathcal{D}| \leq |\mathfrak{S}|^2$, this is $\mathbf{O}(|S|^2|G_C|^2|G_A|^2)$ in the worst case, and *(ii)* Determining the structures in *Lines 9-26*: This is $\mathbf{O}(|\mathfrak{S}|(|\mathcal{O}_C| + |\mathcal{O}_A|)(|\mathfrak{S}|(|U_C| + |U_A|))$. The result follows by combining the two terms. □

We characterize the candidate FSCs returned by Algorithm 12 as follows.

Proposition 7.3. *Algorithm 12 is sound.*

Proof. This is by construction. The output of the algorithm is a set $\{\mathcal{I}^i_C, \mathcal{I}^i_A\}_{i=1}^W$ such that the resulting GMC for each case has a state that is recurrent and has to be visited infinitely often. This state, by Definition 7.7, belongs to $\varphi - RecSet^{\mathcal{C}^i_C, \mathcal{C}^i_A}$. Moreover, if the algorithm returns a nonempty solution, a solution to Problem 7.1 will exist since the FSCs are proper. □

Algorithm 12 is *suboptimal* since we only consider the most recent observations of the controller and adversary. It is also not complete, since there might be a feasible solution that cannot be determined by the algorithm. If no FSC structures of a particular size is returned by Algorithm 12, a heuristic is to increase the number of states in the controller FSC by one, and run the Algorithm again. Once we obtain proper FSC structures of fixed sizes, we will show in later that the satisfaction probability can be improved by adding states to the controller FSC in a principled manner (for adversary FSCs of fixed size). Algorithm 12 and Proposition 7.3 will allow us to restrict our treatment to proper FSCs for the rest of the chapter.

Algorithm 12 Generate candidate FSCs $\mathcal{C}_C, \mathcal{C}_A$

Input: G_C, G_A, $\mathcal{G}_{prod}$, $\mathcal{I}_C^o$, $\mathcal{I}_A^o$
Output: Set of admissible FSC structures $\mathbb{I} := (\mathbb{I}_C, \mathbb{I}_A)$, such that GMC has a $\varphi-$feasible recurrent set
Induce digraph $\mathcal{DG}$ of $\mathcal{M}$ as $(\mathfrak{S}, \mathcal{D})$, s.t. $\forall \mathfrak{s}_1, \mathfrak{s}_2 \in \mathfrak{S} : \mathfrak{s}_1 \to \mathfrak{s}_2 \in \mathcal{D} \Leftrightarrow \bar{P}(\mathfrak{s}_2|\mathfrak{s}_1) > 0$.
$\mathbb{I}_C = \mathbb{I}_A = \emptyset$
$\mathcal{C} = SCCs(\mathcal{DG}) = \{C_1, \ldots, C_N\}$
for $C \in \mathcal{C}$ and $(L_{prod}(i), K_{prod}(i)) \in F_{prod}$ **do**
    $Bad_i = \{\mathfrak{s}' \notin C : \exists \mathfrak{s} \in C$ s.t. $\mathfrak{s}'$ is accessible from $\mathfrak{s}\}$
    $Bad_i = Bad_i \cup (C \cap (L_{prod}(i) \times G_C \times G_A))$
    $Good_i = C \cap (K_{prod}(i) \times G_C \times G_A)$
    Set $\mathcal{I}_C(g'_C, u_C|g_C, o_C) = 1$ for all g'_C, g_C, u_C, o_C
    Set $\mathcal{I}_A(g'_A, u_A|g_A, o_A) = 1$ for all g'_A, g_A, u_A, o_A
    while $\sum_{g'_C, u_C} \mathcal{I}_C(g'_C, u_C|g_C, o_C) > 0 \forall o_C, g_C$ or $\sum_{g'_A, u_A} \mathcal{I}_A(g'_A, u_A|g_A, o_A) > 0 \forall o_A, g_A$ and $Bad_i \neq \emptyset$ **do**
        Choose $\mathfrak{s}' = (s', q', g'_C, g'_A) \in Bad_i$,
$\mathfrak{s}'' = (s'', q'', g''_C, g''_A) \in Good_i$
        for $\mathfrak{s} = (s, q, g_C, g_A) \in C \setminus Bad_i$ **do**
            for $u_C \in U_C$ **do**
                $\mu(g'_C, u_C|g_C, o_C) = \frac{\mathcal{I}_C(g'_C, u_C|g_C, o_C)}{\sum_{g'_C, u_C} \mathcal{I}_C(g'_C, u_C|g_C, o_C)}$
                if $\exists u_A \in U_A$ Eqn. (7.5) holds **then**
                    $\mathcal{I}_C(g'_C, u_C|g_C, o_C) \leftarrow 0$
$\forall g'_C, g_C \in G_C$
                end if
            end for
            for $u_A \in U_A$ **do**
                $\tau(g''_A, u_A|g_A, o_A) = \frac{\mathcal{I}_A(g''_A, u_A|g_A, o_A)}{\sum_{g''_A, u_A} \mathcal{I}_A(g''_A, u_A|g_A, o_A)}$
                if $\forall u_C \in U_C$, Eqn. (7.6) holds **then**
                    $\mathcal{I}_A(g''_A, u_A|g_A, o_A) \leftarrow 0$
                end if
            end for
        end for
        $Bad_i = Bad_i \setminus \{\mathfrak{s}'\}$
    end while
    Construct digraph $\mathcal{DG}_{new}$ of GMC of $\mathcal{G}_{prod}$ under modified $\mathcal{I}_C$ and $\mathcal{I}_A$
    $\mathcal{C}_{new} = SCCs(\mathcal{DG}_{new})$
    if $\exists \mathfrak{s} \in Good_i$ s.t. $\mathfrak{s}$ is *recurrent* in $\mathcal{G}_{new}$ **then**
        $\mathbb{I} = (\mathbb{I}_C \cup \mathcal{I}_C, \mathbb{I}_A \cup \mathcal{I}_A)$
    end if
end for

7.3.3 Value Iteration for POSGs

In this subsection, we present a value-iteration based procedure to maximize the probability of satisfying the LTL formula φ for FSCs $\mathcal{C}_C$ and $\mathcal{C}_A$ of fixed sizes. We prove that the procedure converges to a unique optimal value, corresponding to the Stackelberg equilibrium.

Notice that in Eqn. (7.1), the controller and adversary policies are specified as probability distributions over the next FSC internal state and the respective agent action, and conditioned on the current FSC internal state and the agent observation. We rewrite these in terms of a mapping $\hat{\mu} : G_C \times S \times G_C \times U_C \to [0, 1]$ and $\hat{\tau} : G_A \times S \times G_A \times U_A \to [0, 1]$:

$$\hat{\mu}(g'_C, u_C | g_C, s) := \sum_{o_C \in \mathcal{O}_C} O_C(o_C|s)\mu(g'_C, u_C | g_C, o_C), \tag{7.7}$$

$$\hat{\tau}(g'_A, u_A | g_A, s) := \sum_{o_A \in \mathcal{O}_A} O_A(o_A|s)\mu(g'_A, u_A | g_A, o_A). \tag{7.8}$$

This will allow us to express Equation (7.1) as:

$$\bar{P}((s', q'), g'_C, g'_A | (s, q), g_C, g_A) = \sum_{u_C} \sum_{u_A} \hat{\mu}_C(g'_C, u_C | g_C, s)\hat{\mu}_A(g'_A, u_A | g_A, s) \cdot \bar{P}((s', q'), g'_C, g'_A | (s, q), g_C, g_A) \tag{7.9}$$

Define a value V over the state space of the GMC representing the probability of satisfying the LTL formula φ when starting from a state of the GMC. Additionally, define and characterize the following operators:

$$(T_{\hat{\mu}\hat{\tau}} V)(\mathfrak{s}) = \sum_{\mathfrak{s}'} \bar{P}(\mathfrak{s}'|\mathfrak{s})V(\mathfrak{s}');$$

$$(T_{\hat{\mu}} V)(\mathfrak{s}) = \min_{\hat{\tau}} \sum_{\mathfrak{s}'} \bar{P}(\mathfrak{s}'|\mathfrak{s})V(\mathfrak{s}');$$

$$(TV)(\mathfrak{s}) = \max_{\hat{\mu}} \min_{\hat{\tau}} \sum_{\mathfrak{s}'} \bar{P}(\mathfrak{s}'|\mathfrak{s})V(\mathfrak{s}')$$

where $\bar{P}(\mathfrak{s}'|\mathfrak{s})$ is the transition probability in the GMC induced by policies $\hat{\mu}$ and $\hat{\tau}$ (Eqn. (7.9)). We can characterize value V as follows.

Proposition 7.4. *Let*

$$V((s, q), g_C, g_A) = \max_{\hat{\mu}} \min_{\hat{\tau}} \mathbb{P}(\varphi | ((s, q), g_C, g_A)). \tag{7.10}$$

Then

$$V((s,q),g_C,g_A) = \max_{\hat{\mu}} \min_{\hat{\tau}} \sum_{((s',q'),g'_C,g'_A)} \sum_{u_C} \sum_{u_C} \Bigg(\hat{\mu}(g'_C, u_C | g_C, s) \\ \times \hat{\tau}(g'_A, u_A | g_A, s) P_{prod}((s',q')|(s,q),u_C,u_A) V((s',q'),g'_C,g'_A)) \Bigg) \quad (7.11)$$

Conversely, if the value vector V satisfies Equation (7.11)*, then Eqn.* (7.10) *holds true. Moreover, V is unique.*

Before proving Proposition 7.4, we will need some intermediate results.

Lemma 7.1. *Let V be the satisfaction probability obtained under any pair of policies $\hat{\mu}$ and $\hat{\tau}$, where $\hat{\tau}$ is the best response to $\hat{\mu}$. Let T^k be the operation that composes the T operator k times, and V^k be the corresponding value obtained (i.e., $T^k V := V^k$). Then, there exists a value V^* such that $\lim_{k\to\infty} T^k V = V^*$.*

Proof. We show Lemma 7.1 by showing that the sequence $V^k = T^k V$ is bounded and monotone.

We first show boundedness. By definition of the operator T, V^{k+1} is obtained as a convex combination of V^k. Since V is the satisfaction probability, it is in $[0,1]$. Thus, V^0 is bounded, and consequently, $T^k V$ is bounded for all k.

We next show monotonicity by induction. We have that V^0 is the value function associated with a control policy $\hat{\mu}$. Denote the best response of the adversary to $\hat{\mu}$ as $\hat{\tau}$. Let $V^1 := TV^0$. From the definitions of T and $T_{\hat{\mu}}$, we have $TV^0 \geq T_{\hat{\mu}} V^0$. Furthermore, $V^0 = T_{\hat{\mu}} V^0$ since

$$T_{\hat{\mu}} V^0(\mathfrak{s}) = \min_{\hat{\tau}'} \sum_{\mathfrak{s}'} \sum_{u_C} \sum_{u_A} \Big(V^0(\mathfrak{s}') \hat{\mu}(g'_C, u_C | g_C, s) \hat{\tau}(g'_A, u_A | g_A, s) P_{prod}((s',q')|(s,q),u_C,u_A) \Big) \\ = V^0$$

by the definition that $\hat{\tau}'$ is the best response of $\hat{\mu}$. Therefore, we have that $V^1 = TV^0 \geq T_{\hat{\mu}} V^0 = V^0$. This gives us $V^1 \geq V^0$, which serves as the base case for the induction. Consider iteration k. Suppose $T^{k-1}V \leq T^k V$. We then show $T^k V \leq T^{k+1} V$. We have:

$$T^{k+1}V \geq \min_{\hat{\tau}} \sum_{\mathfrak{s}'} \bar{P}(\mathfrak{s}'|\mathfrak{s}) V^k(\mathfrak{s}') \geq \min_{\hat{\tau}} \sum_{\mathfrak{s}'} \bar{P}(\mathfrak{s}'|\mathfrak{s}) V^{k-1}(\mathfrak{s}') = V^k.$$

The first inequality holds by the definition of T, the second inequality holds by the induction hypothesis that $V^k \geq V^{k-1}$, and the last equality holds by construction of a control policy. The existence of V^* such that $\lim_{k\to\infty} T^k V = V^*$ follows from monotone convergence theorem [119]. □

We now prove Proposition 7.4.

Proof of Proposition 7.4. We first prove the forward direction by contradiction, i.e., if Eqn. (7.10) holds then Eqn. (7.11) holds. Suppose $\hat{\mu}$ is a Stackelberg equilibrium policy with satisfaction probability being V, while Eqn. (7.11) does not hold. Since $\hat{\mu}$ is the Stackelberg equilibrium policy, $V = T_{\hat{\mu}}V$. This is because, given $\hat{\mu}$, the stochastic game is an MDP, for which V is the optimal value [101]. From the definitions of T and $T_{\hat{\mu}}$, we must have $T_{\hat{\mu}}V \leq TV$. Composing the operators k times and letting $k \to \infty$,

$$V = \lim_{k\to\infty} T_{\hat{\mu}}^k V \leq \lim_{k\to\infty} T^k V = V^*,$$

where the first equality holds by the assumption that V is the satisfaction probability and $\hat{\mu}$ is the Stackelberg equilibrium policy, the last equality holds by Lemma 7.1. If $V = V^*$, Eqn. (7.11) is satisfied, which contradicts our assumption that Eqn. (7.10) holds while Eqn. (7.11) does not hold. If $V \neq V^*$, then there must exist a state $\mathfrak{s}$ such that $V(\mathfrak{s}) < V^*(\mathfrak{s})$. This means that there is a policy (different from $\hat{\mu}$) corresponding to V^* for which we achieve a higher satisfaction probability starting at state $\mathfrak{s}$. This violates our assumption that $\hat{\mu}$ is the equilibrium policy. We must then have that Eqn. (7.11) holds given that Eqn. (7.10) holds.

We next prove uniqueness of the V. Let $\hat{V}$ and V be two solutions to Eqn. (7.11), and let $\hat{\mu}$ and μ denote the corresponding control policies. From the definitions of T and $T_{\hat{\mu}}$, we have that $V = TV \geq T_{\hat{\mu}}V$. Composing the operators on both sides k times and letting $k \to \infty$,

$$V = \lim_{k\to\infty} T^k V \geq \lim_{k\to\infty} T_{\hat{\mu}}^k V = \hat{V},$$

where the first equality holds by the assumption that $\hat{\mu}$ is the equilribrium policy, and the second equality holds by the fact that $\hat{V}$ is the unique fixed point of operator $T_{\hat{\mu}}$ [101]. Following a similar argument as before, we have the following inequality:

$$\hat{V} = \lim_{k\to\infty} T^k \hat{V} \geq \lim_{k\to\infty} T_{\hat{\mu}}^k \hat{V} = V.$$

We have that both $V \geq \hat{V}$ and $\hat{V} \geq V$ are true, which gives us $V = \hat{V}$. This implies that the value V is unique.

We finally show that if Eqn. (7.11) holds then Eqn. (7.10) holds. We observe that the value function at equilibrium satisfies (7.11), and the solution is unique. Therefore, any solution to Eqn. (7.11) must be a Stackelberg equilibrium [120]. □

Proposition 7.4 indicates that value-iteration algorithms can be used to determine optimal policies $\hat{\mu}$ and $\hat{\mu}$. Given a policy $\hat{\mu}$ and observation function O_C, we will be able to compute μ by solving a system of linear equations.

A value-iteration based procedure to solve the POSG under an LTL specification is proposed in Algorithm 13. The value $V(\mathfrak{s})$ is greedily updated at every iteration by computing

the policy according to Proposition 7.4. The algorithm terminates when the difference in $V(\cdot)$ in consecutive iterations is below a pre-specified threshold.

Algorithm 13 Maximizing probability of satisfying LTL formula φ under fixed FSCs $\mathcal{C}_C, \mathcal{C}_A$

Input: $\mathcal{M} := \mathcal{M}^{\varphi,\mathcal{C}_C,\mathcal{C}_A}, \{L_{prod}(i), K_{prod}(i)\}_{i=1}^{M}, \epsilon$ (threshold)
Output: $V \in \mathbb{R}^{|S|\times|Q|\times|G_C|\times|G_A|}$
Determine $\varphi - RecSets^{\mathcal{C}_C,\mathcal{C}_A}$ using Algorithm 11
$\hat{\mu}(g'_C, u_C|g_C, s) := \sum_{o_C} O_C(o_C|s)\mu(g'_C, u_C|g_C, o_C)$
$\hat{\tau}(g'_A, u_A|g_A, s) := \sum_{o_A} O_A(o_A|s)\mu(g'_A, u_A|g_A, o_A)$
$V^0(\mathfrak{s}) \leftarrow 0$
$V^1(\mathfrak{s}) \leftarrow 1$ if $\mathfrak{s} \in \varphi - RecSets^{\mathcal{C}_C,\mathcal{C}_A}$, and $V^1(\mathfrak{s}) \leftarrow 0$, else
$k \leftarrow 0$
while $\max_{\mathfrak{s}}\{V^{k+1}(\mathfrak{s}) - V^k(\mathfrak{s})\} > \epsilon$ **do**
    $k \leftarrow k + 1$
    for $\mathfrak{s} \notin \varphi - RecSets^{\mathcal{C}_C,\mathcal{C}_A}$ **do**
        $V^{k+1}(\mathfrak{s}) \leftarrow \max_{\hat{\mu}} \min_{\hat{\tau}} \sum_{\mathfrak{s}'} \sum_{u_C} \sum_{u_A} \Big(V^k(\mathfrak{s}')\hat{\mu}_C(g'_C, u_C|g_C, s)\hat{\tau}(g'_A, u_A|g_A, s) \cdot P_{prod}((s', q')|(s, q), u_C, u_A) \Big)$
    end for
end while
return $V(= V^k(\mathfrak{s}))$

Proposition 7.5. *For any $\epsilon > 0$, there exist K, V such that $\|V^k(\mathfrak{s}) - V\|_\infty < \epsilon$ for all $k > K$. Further, V satisfies the value in Proposition 7.4 and is within the ϵ-neighborhood of the value function at Stackelberg equilibrium.*

Proof. Notice that $V^1(\mathfrak{s}) \geq V^0(\mathfrak{s})$. Since $V^1(\mathfrak{s}) = 0$ for $(s, q) \notin \varphi - RecSets^{\mathcal{C}_C,\mathcal{C}_A}$, $V^2(\mathfrak{s}) \geq V^1(\mathfrak{s})$. We induct on k. Let $\hat{\mu}^k$ be the optimal controller policy at step k. Then,

$$
\begin{aligned}
&V^{k+1}(\mathfrak{s}) \\
\geq &\min_{\hat{\tau}} \sum_{\mathfrak{s}'} \sum_{u_C} \sum_{u_A} \Big(V^k(\mathfrak{s}')\hat{\mu}(g'_C, u_C|g_C, s)\hat{\tau}(g'_A, u_A|g_A, s) P_{prod}((s', q')|(s, q), u_C, u_A) \Big) \\
\geq &\min_{\hat{\tau}} \sum_{\mathfrak{s}'} \sum_{u_C} \sum_{u_A} \Big(V^{k-1}(\mathfrak{s}')\hat{\mu}(g'_C, u_C|g_C, s)\hat{\tau}(g'_A, u_A|g_A, s) P_{prod}((s', q')|(s, q), u_C, u_A) \Big) \\
= &V^k(\mathfrak{s})
\end{aligned}
$$

The first inequality holds because $V^{k+1}(\mathfrak{s})$ is the value obtained by the maximizing policy, and dominates the value achieved by any other policy. The second and last inequalities follow from the induction hypothesis and definition of $V^k(\mathfrak{s})$ respectively. Further, for each state, $V^k(\mathfrak{s})$ is bounded since it is a convex combination of terms that are ≤ 1. Let V be the set of limit points so that K can be chosen such that $\|V^k(\mathfrak{s}) - V\|_\infty < \epsilon$ for $k > K$.

Since $V^k(\mathfrak{s})$ converges, it is a Cauchy sequence. Therefore, for every $\epsilon > 0$, there exists K sufficiently large, such that for all $k > K$, $|V^k(\mathfrak{s}) - V^{k+1}(\mathfrak{s})| < \epsilon$. From *Line 8* of Algorithm 13, this shows that V is within an $\epsilon-$neighborhood of a Stackelberg equilibrium for every $\epsilon > 0$. □

Whenever Algorithm 13 returns a non-empty solution, the FSCs are proper (Definition 7.4). In this case, there is a nonzero probability of visiting a state in a $\varphi-$feasible recurrent set of $\mathcal{G}_{prod}$ under FSCs $\mathcal{C}_C$ and $\mathcal{C}_A$.

7.3.4 Varying the Size of Controller's FSC

In the previous discussions, we assume that the sizes of FSCs for the controller and adversary are fixed. However, it might be possible that the satisfaction probability is nonzero, while Algorithm 13 does not return a non-trivial solution since the sizes of $\mathcal{C}_C$ is not sufficiently large. To this end, we investigate if we can increase the size of the controller's FSC to increase the satisfaction probability. The core idea is that we need to guarantee that adding states to $\mathcal{C}_C$ does not decrease this satisfaction probability when compared to the satisfaction probability for $\mathcal{C}_C$ with fewer states. This lends itself to a policy iteration like approach. Policy iteration [101] is a procedure that alternates between policy evaluation and policy improvement until convergence to a Stackelberg equilibrium. The policy evaluation step involves solving a Bellman equation, while the policy improvement step then 'greedily' chooses a policy that maximizes the satisfaction probability.

In the sequel, we will assume that the size of $\mathcal{C}_A$ is fixed. Intuitively, this means that the controller is aware of the capabilities of the adversary. However, the controller does not know the transition probabilities in $\mathcal{C}_A$, which means it needs to determine the transition probabilities of its own FSC in order to be robust against the worst-case transition probabilities between states in $\mathcal{C}_A$. We will work with the 'value' of a state $g_C \in G_C$ in $\mathcal{C}_C$. Denote this by $V_{g_C}(\mathfrak{s})$. From the controller's perspective, the transition probabilities in $\mathcal{C}_C$ are influenced by its *belief* of the state of $\mathcal{G}_{prod}$ under $\mathcal{C}_C$ and $\mathcal{C}_A$. The *belief* is a (prior) probability distribution over the states of the POSG. Then, the value of $g_C \in G_C$ under belief $b = \{b_1, \ldots, b_{|S|}\}$ can be written as $V_{g_C}(b) = \sum_i b_i V_{g_C}(s_i)$. The value function for a belief b is then given by

$$V(b) := \max_{g_C} \; V_{g_C}(b) \tag{7.12}$$

Fig. 7-1 shows $V(b)$ for a two-state POSG with $|G_C| = 3$.

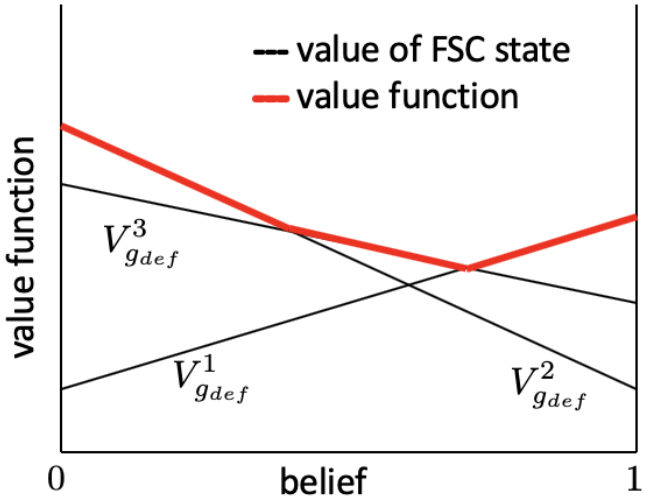


Figure 7-1: Value function of a two-state POSG with three states in $\mathcal{C}_C$. The value of each FSC state is linear in the belief (black lines). The value function is the point-wise maximum of the values of the FSC states (red curve).

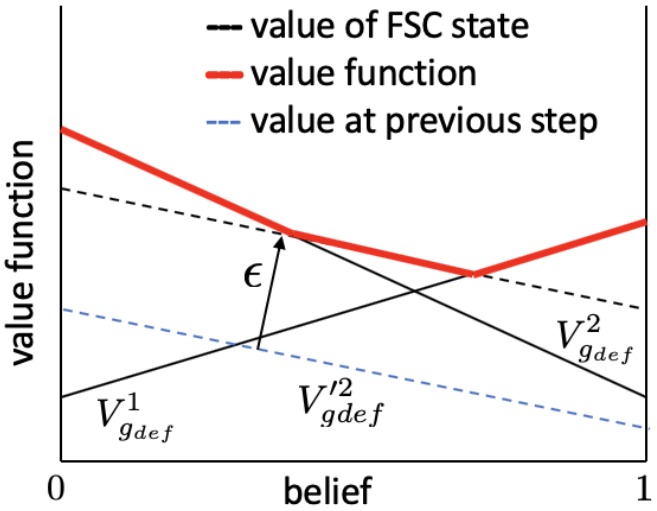


Figure 7-2: Robust linear program for state 2 of $\mathcal{C}_C$. Improved vector $V'^2_{g_C} + \epsilon$ is tangent to the one-step look-ahead value function.

When working with controller and adversary FSCs of fixed sizes, the value iteration in Algorithm 13 terminates when a (local) equilibrium is reached. This means there is no choice of transition probabilities in $\mathcal{C}_C$ that will improve the satisfaction probability for some belief state(s). This probability can be improved by adding states to $\mathcal{C}_C$ (since $|G_A|$ is fixed). The value of a belief b (Eqn. (7.12)) is the point-wise maximum of the value at each state of $\mathcal{C}_C$, which themselves are linear functions of the belief state. Therefore, at equilibrium, $V(b)$ will satisfy:

$$V(b) = \max_{u_C} \min_{u_A} \sum_{o \in \mathcal{O}_C} \mathbb{P}(o|b) V(b_o^{u_C u_A}) \tag{7.13}$$

where

$$\mathbb{P}(o|b) := \sum_s O_C(o|s) b(s), \tag{7.14}$$

$$b_o^{u_C u_A}(s') := \sum_s P(s'|s, u_C, u_A) \frac{O_C(o|s) b(s)}{\sum_{o \in \mathcal{O}_C} O_C(o|s) b(s)} \tag{7.15}$$

This set of equations is not easy to solve since the belief takes values in $[0, 1]$. However, Eqn. (7.13) results in a point-wise improvement of the value function, until an optimum is reached. We will need the following definitions.

Definition 7.8 (Tangent FSC State)**.** *An FSC state g_C is* tangent *to the one-step look-ahead value function $V(b)$ in Equation (7.13) if $V(b) = V_{g_C}(b)$ at state b.*

Definition 7.9 (Improved FSC State)**.** *A state $g_C \in G_C$ is* improved *if transition probabilities associated with that state are changed in a way that increases V_{g_C}.*

For the setting where there are two agents with competing objectives, the problem of determining a policy μ that achieves an improvement in V_{g_C} under any adversary policy τ

can be posed as a robust linear program [121], presented as follows:

$$\max_{\epsilon,\mu,\tau} \quad \epsilon \tag{7.16a}$$

$$\text{s.t.} \quad V_{g_C}(\mathfrak{s}) + \epsilon \leq \sum_{\mathfrak{s}',o,o',g'_C,g'_A,u_C,u_A} \Bigg(V_{g'_C}(\mathfrak{s}') O_C(o|s) \mu(g'_C, u_C|g_C, o) O_A(o'|s) \times \tau(g'_A, u_A|g_A, o') P_{prod}((s',q')|(s,q), u_C, u_A) \Bigg) \forall s, \ \forall \tau(g'_A, u_A|g_A, o') \tag{7.16b}$$

$$\sum_{g', u_C} \mu(g'_C, u_C|g_C, o_C) = 1, \ \forall o_C \tag{7.16c}$$

$$\sum_{g', u_A} \tau(g'_A, u_A|g_A, o_A) = 1, \ \forall o_A \tag{7.16d}$$

$$\mu(g'_C, u_C|g_C, o_C) \in [0,1], \ \forall o_C, g'_C, u_C \tag{7.16e}$$

$$\tau(g'_A, u_A|g_A, o_A) \in \{0,1\}, \ \forall o_A, g'_A, u_A \tag{7.16f}$$

When $\epsilon > 0$, an improvement in the value of the FSC state (by ϵ) can be achieved. This is because there exists a convex combination of value vectors of the one-step look-ahead value function that dominates the present value of the FSC state [122]. The procedure is carried out for each $g_C \in G_C$, until no further improvement in the transition probabilities in $\mathcal{C}_C$ is possible. At this stage, the robust linear program yields $\epsilon = 0$ for every $g_C \in G_C$. The following result generalizes Theorem 2 in [122] to a partially observable environment that includes an adversarial agent.

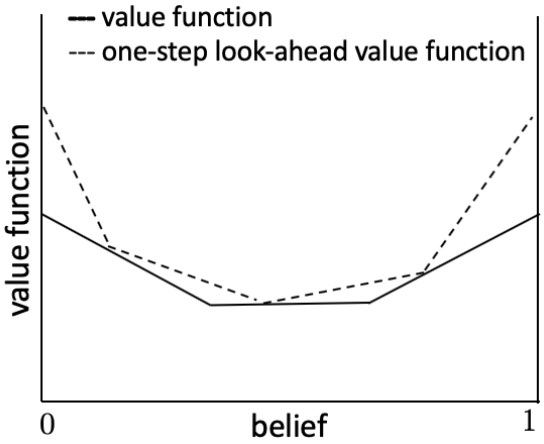


Figure 7-3: At a local equilibrium, all states are tangent to the one-step look-ahead value function.

Proposition 7.6. *The policy iteration procedure has reached a local equilibrium if and only if all the states $g_C \in G_C$ are tangent to $V(b)$.*

Proof. The robust LP aims to maximize the improvement that can be achieved in the value of each state in $\mathcal{C}_C$. From the preceding discussion, and from Definition 7.8, a translation of the value vector of a state g_C by $\epsilon > 0$ will make it tangent to the one-step look-ahead value function. By a similar argument, $\epsilon = 0$ for each $g_C \in G_C$ indicates that improvement in the value of an FSC state will not be possible if it is already tangent to the one-step look-ahead value function. This is shown in Fig. 7-2 and 7-3. □

When $\epsilon = 0$ for each g_C, the satisfaction probability can be improved by adding states to $\mathcal{C}_C$ in a 'principled way'. Let $MaxNewStates$ denote the maximum number of states that can be added to $\mathcal{C}_C$, and let $\{b_k\} := B$ denote the set of belief states satisfying $V(b_k) = V_{g_C}(b)$ for each g_C from Eqn. (7.16a).

Algorithm 14 Adding states to FSC $\mathcal{C}_C$

1: **Input:** Set of belief states $\{b_k\} := B$; $MaxNewStates$
2: **Output:** Set of improved states in FSC $\mathcal{C}_C$
3: $NewStateNumber \leftarrow 0$
4: **while** $B \neq \emptyset$ **do**
5: Choose $b \in B$
6: $B := B \setminus b$
7: $Ahead := \emptyset$
8: **for** $(u_C, u_A, o_C) \in U_C \times U_A \times \mathcal{O}_C$ **do**
9: **if** $\mathbb{P}(o|b) > 0$ in Eqn. (7.14) **then**
10: Determine $b_o^{u_C u_A}(s')$ from Eqn. (7.15)
11: $Ahead = Ahead \cup \{b_o^{u_C u_A}\}$
12: **end if**
13: **end for**
14: **for** $b_A \in Ahead$ **do**
15: Determine $V(b_a)$ from Eqn. (7.13), (7.14), (7.15)
16: Note maximizers u_C^*, g_C^* (Eqn. (7.13), (7.12))
17: **if** $V(b_A) > V(b)$ and $NewStateNumber < MaxNewStates$ **then**
18: Add state, g_{new} to $\mathcal{C}_C$ with $\mu(g_C^*, u_C^* | g_{new}, o) = 1 \ \forall o \in \mathcal{O}_C$
19: $NewStateNumber \leftarrow NewStateNumber + 1$
20: **end if**
21: **end for**
22: **end while**

Algorithm 14 presents a procedure to add states to the controller FSC to improve the satisfaction probability. Lines $6 - 11$ determine the one-step look-ahead beliefs. A new

state is added to $\mathcal{C}_C$ if the controller 'believes' that the probability of satisfying the LTL formula from these states is higher than that from the current belief state (Lines $13-18$). Lines 13 and 14 respectively form the policy evaluation and policy improvement steps of the policy iteration. Edges from new states in the FSC are directed towards the action and FSC state that maximize the value function $V(b)$ in a deterministic manner (Line 16). Values of probabilities and transitions to and from the added state will be adjusted as other FSC states are improved.

Proposition 7.7. *Algorithm 14 terminates in finite time.*

Proof. This follows from the facts the sets B, $\mathcal{O}_C$, U_C, and U_A have finite cardinality, and the one-step look-ahead values in Eqn. (7.13) are upper bounded. □

The procedure yields a new $\mathcal{C}_C$, from which candidate FSC structures can be found using Algorithm 12. The controller policy to maximize the probability of satisfying the LTL formula under any $\mathcal{C}_A$ of fixed size can be determined by Algorithm 13 or the robust linear program in Eqn. (7.16a).

7.4 Case Study

This section presents an example and experiments that illustrate our approach.

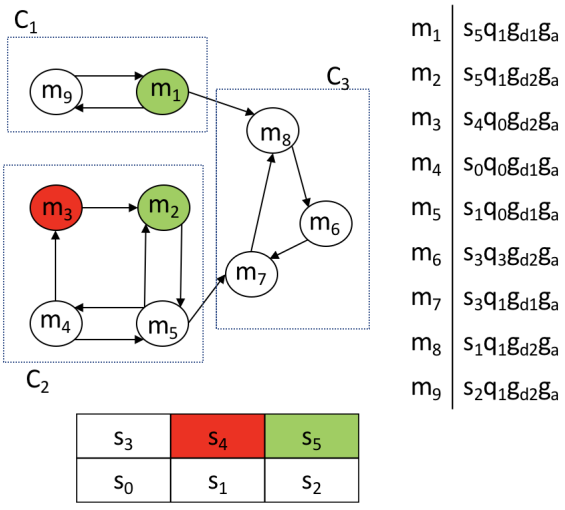
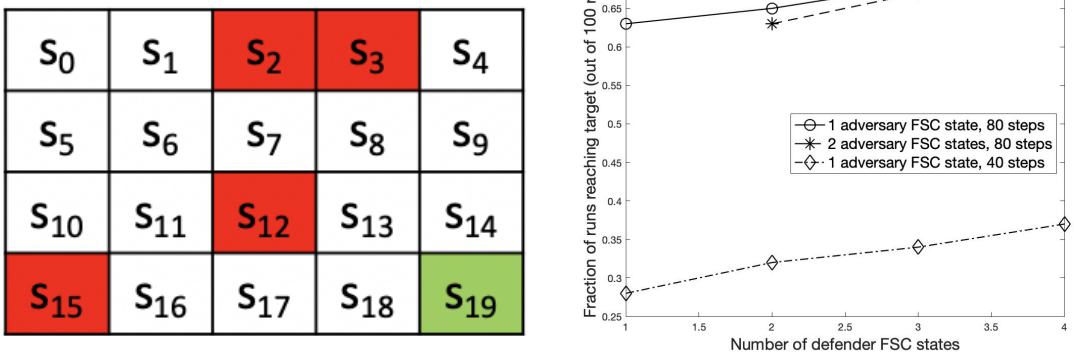


Figure 7-4: Clockwise, from *top-left*: Global Markov chain (GMC) for initial controller and adversary FSC structures- green states (m_1&m_2) must be visited infinitely often, and state in red (m_3) must be visited finitely often in steady-state; GMC state $m_i \in S \times Q \times G_C \times G_A$; State-space for $M = 3, N = 2$ showing unsafe (s_4) and target (s_5) states.



(a) (b)

Figure 7-5: The agent aims to satisfy the LTL formula $\varphi = \square\diamond\texttt{tar} \wedge \square\neg\texttt{obs}$ in the presence of an adversary, in a partially observable environment. The environment is the grid-world in Fig.7-5a. The states in red indicate the presence of an obstacle, the state in green is the target state, and the agent starts in state s_0. The agents' actions are determined by their observations of the state. Assume that the controller FSC has at least as many states as the adversary FSC, i.e. $|G_C| \geq |G_A|$. Fig. 7-5b shows the fraction of runs (out of 100) when the agent reaches the target within 80 steps and 40 steps. This number is higher when $|G_C|$ is larger, for a fixed value of $|G_A|$ ($-\texttt{o}-$, $-*-$, and $-\diamond-$ curves). For the same $|G_C|$, the fraction of successful runs is higher when $|G_A|$ is lower ($-\texttt{o}-$ and $-*-$ curves). The fraction of successful runs is also higher when the agent is allowed more steps to reach the target ($-\texttt{o}-$ and $-\diamond-$ curves).

Consider a state space representing an $M \times N$ grid, $S := \{s_i : i = x + My, x \in \{0, \ldots, M-1\}, y \in \{0, \ldots, N-1\}\}$. The controller's actions are $U_C = \{R, L, U, D\}$ denoting right, left, up, and down, and the actions of the adversary are $U_A = \{A, NA\}$, denoting attack, and not attack respectively. The observations of both agents are $\mathcal{O}_C = \mathcal{O}_A = \{correct, wrong\}$, such that: $O_C(correct|s_i) = 0.8 = 1 - O_C(wrong|s_i)$, and $O_A(correct|s_i) = 0.6 = 1 - O_A(wrong|s_i)$. $\mathcal{O}_C$ and $\mathcal{O}_A$ are probabilities that the agents sense that their observation of the state is indeed the correct state or not. That is, $\mathbb{P}(o_i = s_i)$ or $\mathbb{P}(o_i \neq s_i)$. Let $\Pi = \{\texttt{obs}, \texttt{tar}\}$, denoting obstacle and target respectively. Then, if $\varphi = \square\diamond\texttt{tar} \wedge \square\neg\texttt{obs}$, the corresponding DRA will have two states q_0, q_1, with $F = (\{\emptyset\}, \{q_1\})$. Transition probabilities

for $(u_C, u_A) = (R, NA)$ and (R, A) are defined as:

$$P(s_j|s_i, R, NA) = \begin{cases} 0.8 & j = i+1,\ (i+1) \not\equiv 0 \bmod M \\ \frac{0.2}{|N_{s_i}|} & s_j \in \{s_i\} \cup N_{s_i} \setminus \{s_{i+1}\}),\ (i+1) \not\equiv 0 \bmod M \\ 1 & j = i \text{ and } (i+1) \equiv 0 \bmod M \end{cases}, \tag{7.17a}$$

$$P(s_j|s_i, R, A) = \begin{cases} 0.6 & j = i+1,\ (i+1) \not\equiv 0 \bmod M \\ \frac{0.4}{|N_{s_i}|} & s_j \in \{s_i\} \cup N_{s_i} \setminus \{s_{i+1}\},\ (i+1) \not\equiv 0 \bmod M \\ 1 & j = i \text{ and } i+1 \equiv 0 \bmod M \end{cases}, \tag{7.17b}$$

where N_{s_i} denotes the neighbors of s_i.

Notice that in the above equations, the probability of the agent moving to the 'correct' next state for a particular controller action is larger for the adversary action NA than for the adversary action A. Further, in this case, if the controller is in a square along the right edge of the grid, then the action R does not result in a change of state. The probabilities for other action pairs can be defined similarly.

For this example, let $M = 3$ and $N = 2$. Then, $|S| = 6$. Let s_4 be an unsafe state, and s_5 be the goal state. This is indicated in Fig. 7-4. Let $|G_C| = 2$, $|G_A| = 1$ for the FSCs. Assume that for some initial structures $\mathcal{I}_C^0, \mathcal{I}_A^0$ the GMC is given by Fig. 7-4. The figure also indicates the states in terms of its individual components. Assume that the LTL formula φ is such that the states in green denote those that have to be visited infinitely often in steady state, while those in red must be avoided. Therefore $(L_{prod}, K_{prod}) = \{(\{\emptyset\}, \{m_1\}), (\{m_3\}, \{m_2\})\}$. The boxes C_1, C_2, C_3 indicate the communicating classes of the graph.

From Algorithm 12, for C_1, $Bad = \{m_8\}, Good = \{m_1\}$. For $m_1 \to m_8$, Eqn. (7.5) is true for all u_A and $u_C = \{D, L\}$. Thus, $\mathcal{I}_C(g', u_C|g, o) \leftarrow 0$ for $o = \{correct, wrong\}$. For $m_9 \to m_1$, since Eqn. (7.6) does not hold for $R, D \in U_C$, $\mathcal{I}_A(\cdot)$ is unchanged. Then, m_1 is recurrent in $\mathcal{G}_{new}$. For C_2, $Bad = \{m_3, m_7\}, Good = \{m_2\}$. Like for C_1, $\mathcal{I}_A(\cdot)$ remains unchanged, since Eqn. (7.6) does not hold for $D \in U_C$. For $m_5 \to m_7$, $\mathcal{I}_C(g', u_C|g, o) \leftarrow 0 \forall u_C \in U_C \setminus D$. A similar conclusion is drawn for $m_4 \to m_3$. Then, m_2 will be recurrent in $\mathcal{G}_{new}$. For C_3, since $Bad = Good = \emptyset$, no structure is added to $\mathcal{I}$. Notice that these FSCs satisfy Proposition 7.1.

This example also shows the limitations of Algorithm 12. From the $M \times N$ grid, there is a policy that takes the controller from any $s \in S \setminus \{s_4\}$ to s_5 with probability 1. However, for FSCs of small size, the initial state of the controller might result in the Algorithm reporting that no solution was found, even if there exists a feasible solution.

We now suppose that $M = 5$ and $N = 4$. A representation of the environment is shown in Fig. 7-5a. The LTL formula to be satisfied is $\varphi = \square\diamond\texttt{tar} \wedge \square\neg\texttt{obs}$. The observation function of the defender is modified so that for a state s where $\mathcal{L}(s) = \texttt{obs}$ or $\mathcal{L}(s) = \texttt{tar}$, $O_C(correct|s) = 1$. That is, the defender recognizes an obstacle or the target correctly with probability one. Our experiments compute the probability of reaching the target under limited sensing capabilities of the agents with FSCs having different number of states. In

each case, we assume that $|G_C| \geq |G_A|$. The number of states in the GMC varies from 60 to 480. Table 7.1 shows the average satisfaction probability and standard deviation of

$\|\mathbf{G_C}\|$	$\|\mathbf{G_A}\| = \mathbf{1}$: $\mathbf{V(\mathfrak{s}_0)}$ **[Std. Dev.]**	$\|\mathbf{G_A}\| = \mathbf{2}$: $\mathbf{V(\mathfrak{s}_0)}$ **[Std. Dev.]**
1	0.53 [0.04]	0.45 [0.05]
2	0.55 [0.05]	0.45 [0.06]
3	0.56 [0.09]	0.47 [0.08]
4	0.57 [0.10]	0.48 [0.12]

Table 7.1: Satisfaction probability and standard deviation (over 100 trials) of reaching the target state s_{19} from s_0 for LTL formula $\varphi = \square\diamond\texttt{tar} \wedge \square\neg\texttt{obs}$ starting from s_0 for varying number of defender FSC states $|G_C|$, when number of states in adversary FSC, $|G_A| = 1$ and $|G_A| = 2$.

reaching the target state s_{19} starting from s_0 (expressed in terms of the value of the state from Eqn. (7.10)) when the adversary FSC has one and two states. Higher values of the standard deviation could be due to the fact that in some cases, Algorithm 12 may terminate before $V(\mathfrak{s}_0)$ is updated enough number of times.

$\|\mathbf{G_C}\|$	**Benign Baseline**	**Adv. Baseline**	**Adv.-Aware (ours)**
1	0.69	0.35	**0.53**
2	0.70	0.35	**0.55**
3	0.73	0.38	**0.56**
4	0.75	0.39	**0.57**

Table 7.2: Comparison of probabilities of satisfying LTL formula $\varphi = \square\diamond\texttt{tar} \wedge \square\neg\texttt{obs}$ starting from s_0 in the presence and absence of an adversary. The first column lists the number of defender FSC states. Subsequent columns enumerate satisfaction probabilities in the following scenarios: i) absence of adversary (Benign Baseline [1]); ii) using a defender policy that was synthesized without an adversary, but realized in the presence of an adversary (Adversarial Baseline); iii) using a defender policy designed assuming the presence of an adversary (Adversary-Aware Design- **our approach**). Although the benign baseline gives the highest satisfaction probability, the same baseline when used in the presence of an adversary results in a much lower satisfaction probability. In comparison, our *Adversary-Aware Design* approach results in a higher satisfaction probability than the 'Adversarial Baseline' where we use a defender policy designed to account for adversarial behavior. We assume that the adversary FSC has one state, so that the GMC with and without the adversary FSC will have the same number of states.

Table 7.2 compares the probabilities of satisfying the LTL objective $\varphi = \square\diamond\texttt{tar}\wedge\square\neg\texttt{obs}$

starting from s_0 in the presence and absence of an adversary. We compare our approach with a baseline, which is a defender policy synthesized in the absence of an adversary. The GMC for the case without an adversary is constructed using the approach of [1]. This baseline defender policy in then realized in the presence of an adversary, and we compare it with our method of synthesizing a defender policy assuming the presence of an adversary. Although the highest satisfaction probability is got while using the baseline policy, when this baseline is used in the presence of an adversary, we obtain a much lower satisfaction probability. In comparison, our *adversary-aware defender policy* results in a higher satisfaction probability than when using the baseline in the presence of the adversary. We note that for this comparison, the adversary FSC has one state, i.e., $|G_A| = 1$, so that the GMCs with and without the adversary FSC have the same number of states.

Fig. 7-5b shows the fraction of sample paths when the agent reaches the target for the first time. After this, since the agent is in a recurrent set, it will continue to visit states in this set with probability one. The following observations can be drawn from Fig. 7-5b. First, for a fixed $|G_A|$, the fraction of runs when the agent successfully reaches the target increases as $|G_C|$ increases. Second, for a fixed $|G_C|$, the probability of satisfying φ is higher for a smaller $|G_A|$. Third, the fraction of successful runs improves with allowing the agent more steps to reach the target. One reason for the first two observations is that the number of states in an FSC models the 'memory' available to the agent. The defender can play better when it has more FSC states, or when the adversary has fewer FSC states. While our results agree with intuition, a caveat is that these numbers depend on the agents' observations, $\mathcal{O}_C$ and $\mathcal{O}_A$. Here, the observations of the agents is an indication of whether the state is the actual state the defender is in. This could be a reason for fewer successful runs when the agent is allowed a maximum of 40 steps versus the case when it is allowed 80 steps.

7.5 Conclusion

This chapter investigated the problem of synthesizing a policy which is represented by a finite state controller to maximize the probability of satisfying an LTL formula in an adversarial environment with partial observability. We proposed to use partially observable stochastic game (POSG) to represent the interaction between the controller and adversary. The finite state controllers were composed with the POSG to yield a fully observable policy-induced Markov chain, named global Markov chain (GMC). We showed that the probability of satisfaction of the LTL formula was equal to the probability of reaching recurrent classes of this MC. We subsequently presented a procedure to determine defender and adversary controllers of fixed sizes that result in nonzero satisfaction probability of the LTL formula, and proved its soundness. Maximizing the satisfaction probability was related to reaching a Stackelberg equilibrium of a stochastic game involving the agents through a value-iteration based procedure. Finally, we showed a means to add states to the defender FSC in a principled way in order to improve the satisfaction probability for adversary FSCs of fixed sizes.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 8

Control Synthesis under Time-Critical Constraints in the Presence of Timing and Actuator Attacks

8.1 Introduction

Chapter 4 to 7 focus on linear temporal logic, which specifies the order of the event that should occur. In some applications, we need to assign time-critical specifications to CPS, which normally involve deadlines and time intervals, making LTL specifications less effective. One way to express time-critical properties is to use Metric Interval Temporal Logic (MITL) [123]. MITL uses intervals of length larger than zero to augment timing constraints to modalities of LTL.

Although MITL enables us to specify the timed behavior of CPS, incorporating timing constraints also introduces an additional attack surface to the malicious adversary. An intelligent adversary can exploit the vulnerabilities and alter the timed behavior of the system, thereby causing violation of the MITL specification. Hence, we need to study an adversary that have abilities to launch attacks on the clocks of the system (*timing attack*) or tamper with inputs to the system (*actuator attack*). A timing attack will prevent the system from reaching desired states within the specified time interval. An actuator attack will allow the adversary to steer the system away from a target set of states.

In this chapter, we investigate how to synthesize a control policy for CPS in the presence of a malicious adversary that is capable of launching timing and actuator attacks such that the probability of satisfying a given MITL specification is maximized. This problem has two major challenges: (i) stochastic games are not sufficient to model the interaction between the controller and adversary, and (ii) the controller could incorrectly perceive the time index

that it observes if it is the target of a timing attack, leading to an incomplete information for the controller. To this end, we make the following contributions in this chapter:

- We define a new entity called a *durational stochastic game (DSG)* that captures both time-sensitive objectives and the presence of an adversary.
- We construct the controller policy using *finite state controllers (FSCs)*. This will allow it to satisfy the MITL objective in cases of timing attacks. The states of the FSC correspond to the difference between values of the estimated and observed time indices.
- We prove that satisfying the MITL formula is equivalent to reaching a subset of states of a *global DSG* constructed by composing representations of the MITL objective, CPS under attack, and FSC. We give a computational procedure to determine this set.
- We develop a value-iteration based algorithm that maximizes the probability of satisfying the MITL formula for FSCs of fixed sizes under any adversary policy.
- We evaluate our approach on a representation of a signalized traffic network. The adversary is assumed to have the ability to mount actuator and timing attacks on the traffic signals. Our numerical results indicate a significant improvement in the probability of satisfying the given MITL specification compared to two baselines.

The remainder of this chapter is organized as follows. Section 8.2 introduces the system model, adversary model, and presents the problem formulation on maximizing the probability of satisfying a given MITL specification in the presence of timing and actuator attacks. Section 8.3 presents our proposed solution approach. Section 8.4 evaluates our proposed approach on a signalized traffic network and a two-tank system, and Section 8.5 concludes this chapter.

8.2 System Model and Problem Statement

This section introduces the adversary and controller models. We then present an entity called a durational stochastic game (DSG) that models the interaction between the controller and adversary. The DSG also models the possible amount of time taken for a transition between two states to be completed. We end the section by formally stating the problem that this chapter seeks to solve.

We consider a CPS whose dynamics is given as

$$x(k+1) = f(x(k), u_C(k), u_A(k), w(k)), \tag{8.1}$$

where k is the time index, $x(k)$ is the state of the system, $u_C(k)$ and $u_A(k)$ are the controller's and adversary's inputs, and $w(k)$ is a stochastic disturbance. The time index starts at $k = 0$,

and is known to both players. The initial state $x(0)$ and statistical information of $w(k)$ is also known to both players. The controller aims to synthesize a sequence of inputs to maximize the probability of the MITL objective φ being satisfied. The adversary aims to reduce this probability.

8.2.1 Adversary and Controller Models

The adversary can launch an *actuator attack* or a *timing attack*, or a combination of the two to achieve its objective.

During an actuator attack, the adversary manipulates control signals received by the actuator. The sequence of inputs supplied by the adversary in this case is called the *actuator attack policy*, denoted τ. This attack can be effected when the controller communicates with the actuator via an unreliable communication channel. In Eqn. (8.1), the adversary can tamper with the control input $u_C(k)$ by injecting a signal $u_A(k)$. Then, the transition of the system to the next state will be jointly determined by $u_C(k)$ and $u_A(k)$.

To launch a timing attack, an adversary can target the time synchronization protocol of the controller [124, 125]. This will affect the controller's perception of the (correct) time index. The sequence of inputs supplied by the adversary in this case is called the *timing attack policy*, denoted ξ. The adversary manipulates time stamps k associated with measurements made by the controller as $k + \kappa$, where $\kappa \geq -k$ is an integer. The policies τ and ξ will be defined later in this section.

At each time k, the adversary can observe the state $x(k)$ and the correct time index k. The observation made by the adversary at time k is defined as $Obs_A^k := \{x(k), k\}$. The adversary also knows the policy (sequence of inputs) μ committed to by the controller. Thus, the overall information $\mathcal{I}_A$ available to the adversary is $\mathcal{I}_A := \bigcup_{m=0:k} Obs_A^m \cup \{\mu\}$.

Different from the information available to the adversary, the controller observes the system state x and a time k', i.e., $Obs_C^{k'} := \{x(k'), k'\}$, where $x(k') = x(k)$ is the state measurement at time k with possibly incorrect time stamp k' due to a timing attack by the adversary. The overall information available to the controller is $\mathcal{I}_C := \bigcup_{m=0:k} Obs_C^m$. Later in this chapter, we will use finite state controller (FSC) as a formal representation of μ.

8.2.2 Durational Stochastic Game

We present an abstraction of the CPS (8.1), that we term a durational stochastic game (DSG). A DSG models the interaction between the controller and adversary, and captures the time taken for a state transition. Let Δ be a discrete set of possible amounts of time taken for a transition between two states in the DSG, given specific agent actions. We then define DSG as follows.

Definition 8.1 (Durational Stochastic Game)**.** *A (labeled) durational stochastic game (DSG) is a tuple* $\mathcal{G} = (S_\mathcal{G}, s_{\mathcal{G},0}, U_C, U_A, Inf_{\mathcal{G},C}, Inf_{\mathcal{G},A}, P_\mathcal{G}, T_\mathcal{G}, \Pi, L, C)$*, where*

- $S_{\mathcal{G}}$ *is a finite set of states,*
- $s_{\mathcal{G},0}$ *is the initial state.*
- U_C *and* U_A *are finite sets of actions for the controller and adversary, respectively.*
- $Inf_{\mathcal{G},C}$ *and* $Inf_{\mathcal{G},A}$ *are the information sets of the controller and adversary, respectively.*
- $P_{\mathcal{G}} : S_{\mathcal{G}} \times U_C \times U_A \times S_{\mathcal{G}} \to [0,1]$ *encodes* $P_{\mathcal{G}}(s'_{\mathcal{G}}|s_{\mathcal{G}}, u_C, u_A)$*, the transition probability from state* $s_{\mathcal{G}}$ *to* $s'_{\mathcal{G}}$ *when the controller and adversary take actions* u_C *and* u_A*.*
- $T_{\mathcal{G}} : S_{\mathcal{G}} \times U_C \times U_A \times S_{\mathcal{G}} \times \Delta \to [0,1]$ *is a probability mass function.* $T_{\mathcal{G}}(\delta|s_{\mathcal{G}}, u_C, u_A, s'_{\mathcal{G}})$ *denotes the probability that a transition from* $s_{\mathcal{G}}$ *to* $s'_{\mathcal{G}}$ *under actions* u_C *and* u_A *takes* $\delta \in \Delta$ *time units.*
- Π *is a set of atomic propositions.*
- $L : S_{\mathcal{G}} \to 2^{\Pi}$ *is a labeling function that maps each state to atomic propositions in* Π *that are true in that state, and* C *is the set of clocks.*

In this chapter, we assume the transition probability $P_{\mathcal{G}}$ and probability mass function $T_{\mathcal{G}}$ are known to both the controller and adversary. In Definition 8.1, the transition probability between states is jointly determined by actions taken by the controller and adversary, which models an actuator attack. The asymmetry of information sets of the two agents models a timing attack. This can be justified as follows: let the actions available to the agents at a state $s \in S_{\mathcal{G}}$ be $U_C(s)$ and $U_A(s)$, and let the respective information sets be $Inf_{\mathcal{G},C}(s)$ and $Inf_{\mathcal{G},A}(s)$. In order to capture the information pattern described in Section 8.2.1, we have $Inf_{\mathcal{G},C}(s) = \{(s_0, \mathbf{v}_0), \cdots, (s, \bar{\mathbf{v}})\}$, i.e., the controller knows the path from the initial state s_0 to current state s along with the time stamp of each state being reached. Here $\mathbf{v}$ denotes the clock valuation (see Chapter 3 for its definition). We reiterate that the time stamps observed by the controller could have been manipulated by the adversary, and hence may be incorrect. The adversary knows the path from the initial state s_0 to current state s along with the correct time stamps of each state being reached, and the controller policy, i.e., $Inf_{\mathcal{G},A}(s) = \{(s_0, \mathbf{v}_0), \cdots, (s, \mathbf{v})\} \cup \{\mu\}$.

For the remainder of this chapter, we use the DSG $\mathcal{G}$ as an abstraction of the CPS (8.1). The mapping from the CPS model (8.1) to a DSG is presented in Algorithm 15. Algorithm 15 partitions the state space and the admissible control and adversary action sets (lines 5-6). We use Monte-Carlo simulation [126] to compute the transition probability distributions $P_{\mathcal{G}}$ and and $T_{\mathcal{G}}$ (lines 8-17).

8.2.3 Problem Statement

Similar to Chapter 4 to 7, we consider the Stackelberg setting between the controller and adversary, with the controller acts as the leader and the adversary being the follower. In

Algorithm 15 Constructing a DSG abstraction for CPS.

Input: CPS model $f(x(k), u_C(k), u_A(k), w(k))$
Output: DSG $\mathcal{G}$
Initialize time-horizon K
Partition the state space as $\mathcal{X} = \cup_{i=1}^{n} X_i$
Partition control and adversary input as sets of polytopes $U_C = \{u_{C_1}, \cdots, u_{C_\Xi}\}$, $U_A = \{u_{A_1}, \cdots, u_{A_\Gamma}\}$
$S = \{X_1, \ldots, X_n\}$ and $\mathcal{L}$ is determined accordingly
for $l = 1, \ldots, n$ **do**
    for all $u_C \in U_C$ and $u_A \in U_A$ **do**
        for $k = 1, \ldots, K$ **do**
            $x \leftarrow$ sampled state in X_i
            $\hat{u}_C, \hat{u}_A \leftarrow$ sampled inputs from u_C, u_A
            $j \leftarrow$ region containing $f(x, \hat{u}_C, \hat{u}_A, \vartheta)$
            Use particle filter to approximate transition probabilities $P_{\mathcal{G}}$ and duration function $T_{\mathcal{G}}$ between sub-regions X_i and X_j for all i and j.
        end for
    end for
end for

this chapter, we use finite state controllers to represent the policy of the controller. For the time-being, however, it will suffice to think of the controller's policy as a probability distribution over the controller actions, given the state of DSG. Let V be a finite set of clock valuations (see Chapter 3 for its definition). The adversary policies corresponding to the two types of attacks is formally stated below.

Definition 8.2 (Adversary policies)**.** *The* actuator attack policy *is a map $\tau : S_{\mathcal{G}} \times V \mapsto U_A$. That is, τ specifies an action $u_A \in U_A(s)$ for each state $(s, \mathbf{v}) \in S$.*

The timing attack policy *is a map $\xi : V \times V \mapsto [0, 1]$. That is, ξ encodes $\xi(\mathbf{v}'|\mathbf{v})$, the probability that the adversary will manipulate the correct clock valuation $\mathbf{v}$ to a valuation $\mathbf{v}'$.*

We are now ready to state the problem.

Problem 8.1. *Given an MITL objective φ, and a DSG $\mathcal{G}$ in which the controller's objective is to maximize the probability of satisfying φ and the adversary's objective is to minimize this probability, compute a control policy that is in Stackelberg equilibrium, i.e.,*

$$\max_{\mu} \min_{\tau, \xi} \mathbb{P}(\varphi). \tag{8.2}$$

8.3 Control Synthesis under MITL Constraints in the Presence of Timing and Actuator Attacks

This section presents the our proposed solution approach. We first compute a product durational stochastic game (PDSG), given a DSG that abstracts the CPS, and a TBA corresponding to the MITL formula φ. We represent the controller's policy as a finite state controller (FSC), and compute a global DSG (GDSG) by composing the PDSG and FSC. We solve Problem 8.1 by proving that maximizing the probability of satisfying φ is equivalent to maximizing the probability of reaching a subset of states of the GDSG, termed *generalized accepting end components (GAMECs)*. Then, we present a value-iteration based algorithm to synthesize an FSC that will lead to an SE of the game between controller and adversary.

8.3.1 Product Durational Stochastic Game Construction

We construct an entity named product durational stochastic game (PDSG) as follows.

Definition 8.3 (Product Durational Stochastic Game)**.** *A PDSG $\mathcal{P}$ constructed from a DSG $\mathcal{G}$, a TBA $\mathcal{A}$, and clock valuation set V is a tuple $\mathcal{P} = (S, s_0, U_C, U_A, Inf_C, Inf_A, P, Acc)$, where*

- *The set $S = S_{\mathcal{G}} \times Q \times V$ is a finite set of states.*
- *$s_0 = (s_{\mathcal{G},0}, q_0, \mathbf{v}_0)$ is the initial state.*
- *U_C and U_A are finite sets of actions for the controller and adversary, respectively.*
- *Inf_C and Inf_A are the information sets of the controller and adversary, respectively.*
- *$P : S \times U_C \times U_A \times S \mapsto [0, 1]$ encodes $P((s', q', \mathbf{v}')|(s, q, \mathbf{v}), u_C, u_A)$, the probability of a transition from state $(s, q, \mathbf{v})$ to $(s', q', \mathbf{v}')$ when the controller and adversary take actions u_C and u_A respectively. The probability*

$$P\left((s', q', \mathbf{v}')|(s, q, \mathbf{v}), u_C, u_A\right) \coloneqq T_{\mathcal{G}}(\delta|s, u_C, u_A, s')P_{\mathcal{G}}(s'|s, u_C, u_A) \tag{8.3}$$

  if and only if $(q, \mathbf{v}) \xrightarrow{L(s'),\delta} (q', \mathbf{v}')$.

- *$Acc = S_{\mathcal{G}} \times F \times V$ is a finite set of accepting states.*

At a state $(s, q, \mathbf{v}) \in S$, let $Inf_C(s, q, \mathbf{v}) := \{(s_0, q_0, \mathbf{v}_0), \cdots, (s, q, \bar{\mathbf{v}})\}$ (the controller knows the path from the initial state of PDSG to the current state, along with the manipulated time stamps) and $Inf_A(s, q, \mathbf{v}) := \{(s_0, q_0, \mathbf{v}_0), \cdots, (s, q, \mathbf{v})\} \cup \{\mu\}$ (the adversary knows the controller's policy μ and the path from the initial state to the current state, along with the correct time stamps).

The following result establishes the consistency of the PDSG $\mathcal{P}$.

Proposition 8.1. *The function $P(\cdot)$ is well-defined. That is, $P\left((s', q', \mathbf{v}')|(s, q, \mathbf{v}), u_C, u_A\right) \in [0, 1]$ and*

$$\sum_{(s', q', \mathbf{v}')} P\left((s', q', \mathbf{v}')|(s, q, \mathbf{v}), u_C, u_A\right) = 1. \tag{8.4}$$

Proof. For any transition in $\mathcal{P}$, $P\left((s', q', \mathbf{v}')|(s, q, \mathbf{v}), u_C, u_A\right) \in [0, 1]$. This is due to the fact that $T_{\mathcal{G}}(\delta|s, u_C, u_A, s') \in [0, 1]$ and $P_{\mathcal{G}}(s'|s, u_C, u_A) \in [0, 1]$. Moreover, we have that: (i) $P\left((s', q', \mathbf{v}')|(s, q, \mathbf{v}), u_C, u_A\right) = 0$ iff $T_{\mathcal{G}}(\delta|s, u_C, u_A, s') = 0$, or $P_{\mathcal{G}}(s'|s, u_C, u_A) = 0$, or both, and (ii) $P\left((s', q', \mathbf{v}')|(s, q, \mathbf{v}), u_C, u_A\right) = 1$ iff $T_{\mathcal{G}}(\delta|s, u_C, u_A, s') = 1$ and $P_{\mathcal{G}}(s'|s, u_C, u_A) = 1$. Let $I^{\delta}_{(q,\mathbf{v}),(q',\mathbf{v}')} := \mathbb{1}((q, \mathbf{v}) \xrightarrow{L(s'),\delta} (q', \mathbf{v}'))$ be an indicator function that takes value 1 if its argument is true, and 0 otherwise. Then, Eqn. (8.4) can be rewritten as:

$$\sum_{(s', q', \mathbf{v}')} T_{\mathcal{G}}(\delta|s, u_C, u_A, s') P_{\mathcal{G}}(s'|s, u_C, u_A) \tag{8.5}$$

$$= \sum_{s' \in S_{\mathcal{G}}} \sum_{\delta \in \Delta} T_{\mathcal{G}}(\delta|s, u_C, u_A, s') I^{\delta}_{(q,\mathbf{v}),(q',\mathbf{v}')} P_{\mathcal{G}}(s'|s, u_C, u_A) \tag{8.6}$$

$$= 1, \tag{8.7}$$

where Eqn. (8.5) holds by substituting from Equation (8.3), Eqn. (8.6) follows from Definition 8.3 and $P_{\mathcal{G}}(s'|s, u_C, u_A) > 0$, and Equation (8.7) results by observing that $\sum_{\delta \in \Delta} T_{\mathcal{G}}(\delta|s, u_C, u_A, s') = 1$ and $\sum_{s' \in S_{\mathcal{G}}} P_{\mathcal{G}}(s'|s, u_C, u_A) = 1$. □

From Eqn. (8.3), we observe that a transition exists in $\mathcal{P}$ if and only if the label associated with the target state matches the atomic proposition corresponding to the transition in the TBA, and the clock constraint is satisfied. Further, for any run $\beta := (s_0, q_0, \mathbf{v}_0), (s_1, q_1, \mathbf{v}_1), \ldots$ on $\mathcal{P}$, we can obtain a run ρ on $\mathcal{A}$ and a path on $\mathcal{G}$. That is, there is a one-one mapping from runs on the PDSG to those on the TBA and DSG. We define the following two projections over the runs on $\mathcal{P}$. Given a run β, we let $\mathsf{Untime}(\beta) = (s_0, q_0), (s_1, q_1), \cdots$, be the untimed sequence of states, and let $\mathsf{Time}(\beta) = (q_0, \mathbf{v}_0), (q_1, \mathbf{v}_1), \cdots$, be the configuration sequence corresponding to β.

8.3.2 Controller Policy Representation: Finite State Controllers

We now formally define the controller's policy μ. Since the adversary can manipulate the clock valuation $\mathbf{v}$ observed by the controller, the controller has only partial information over the DSG. This is evident from the following: let there exist a run $\beta = (s_0, q_0, \mathbf{0})(s_1, q_1, \mathbf{1})(s_2, q_2, \mathbf{2})$ on PDSG $\mathcal{P}$ without any clock being reset that is manipulated by the adversary as $\beta' = (s_0, q_0, \mathbf{0})(s_1, q_1, \mathbf{1})(s_2, q_2, \mathbf{0.5})$. The run β' is not reasonable since the time sequence $\mathsf{Time}(\beta') = (q_0, \mathbf{0}), (q_1, \mathbf{1}), (q_2, \mathbf{0.5})$ is not monotone. The presence of such a run will allow the controller to conclude that a timing attack has been effected by the adversary. Moreover, after a timing attack has been detected, the controller will be

aware that the observed clock valuation is incorrect, and thus cannot be relied upon for control synthesis. The controller will then need to keep track of an estimate of the clock valuation in order to detect a timing attack, and use this estimate for control synthesis. The controller's policy is represented as a finite state controller (FSC) defined as follows.

Definition 8.4 (Finite State Controller [127]). *A finite state controller (FSC) is a finite state automaton $\mathcal{F} = (Y, y_0, \mu)$, where $Y = \Lambda \times \{0, 1\}$ is a finite set of internal states, Λ is a set of estimates of clock valuations, the set $\{0, 1\}$ indicates if a timing attack has been detected (1) or not (0). y_0 is the initial internal state. μ is the controller policy, given by:*

$$\mu = \begin{cases} \mu_0 : Y \times S \times Y \times U_C \mapsto [0, 1], \text{ if } \mathcal{H}_0 \text{ holds}; \\ \mu_1 : Y \times S_{\mathcal{G}} \times Q \times Y \times U_C \mapsto [0, 1], \text{ if } \mathcal{H}_1 \text{ holds}, \end{cases} \tag{8.8}$$

where μ_0 and μ_1 respectively denote the control policies that will be executed when hypothesis $\mathcal{H}_0$ or $\mathcal{H}_1$ holds.

In Definition 8.4, the hypothesis $\mathcal{H}_0$ models the scenario where no timing attack has been detected by the controller, and $\mathcal{H}_1$ models the case when a timing attack has been detected. Eqn. (8.8) specifies the probability of reaching the next internal state y' and taking the corresponding action u_C, given the current internal state y, observed clock valuation $\mathbf{v}$ (if no timing attack has been detected), and state s of $\mathcal{G}$. The FSC allows the controller to synthesize policies with finite memory rather than memoryless policies. In this chapter, we assume the size of the FSC is given and fixed and limit our focus to computing μ. The two players can track an estimate of the clock valuation according to the probability distribution $T_{\mathcal{G}}$. Moreover, we do not explicitly specify a timing attack detection scheme, and assume it is known. Timing attack detection schemes that are compatible with our framework include [124] and [125]. In the nominal case, the controller adopts the policy μ_0. Once a timing attack has been detected by the controller, the controller ignores the observed clock valuation $\mathbf{v}$, and switches to policy μ_1. The design of a timing attack detection strategy is beyond the scope of this chapter, and we leave it as future work.

8.3.3 Value Iteration Algorithm for Control Synthesis

To incorporate the evolution of the estimate of the clock valuation maintained by the controller, we compose this with the PDSG. We call this entity the global DSG (GDSG). We prove that maximizing the probability of satisfying the MITL objective φ is equivalent to maximizing the probability of reaching a specific subset of states in the GDSG called generalized accepting maximal end components (GAMECs). The control policy is then computed using a value iteration based procedure. Given an FSC $\mathcal{F}$ and the PDSG $\mathcal{P}$, we can construct GDSG in the following way.

Definition 8.5 (Global DSG (GDSG)). *A GDSG is a tuple $\mathcal{Z} = (S_{\mathcal{Z}}, s_{\mathcal{Z},0}, U_C, U_A, Inf_{\mathcal{Z},C}, Inf_{\mathcal{Z},A}, P_{\mathcal{Z}}, Acc_{\mathcal{Z}})$, where*

- $S_{\mathcal{Z}} = S \times Y$ *is a finite set of states.*
- $s_{\mathcal{Z},0} = (s_0, q_0, \mathbf{v}_0, y_0)$ *is the initial state.*
- U_C *and* U_A *are finite sets of actions for the controller and adversary, respectively.*
- $Inf_{\mathcal{Z},C}$ *and* $Inf_{\mathcal{Z},A}$ *are the information sets of the controller and adversary respectively.*
- $P_{\mathcal{Z}} : S_{\mathcal{Z}} \times U_C \times U_A \times S_{\mathcal{Z}} \mapsto [0,1]$ *is a transition function with the probability of a transition from state* $(s, q, \mathbf{v}, y)$ *to* $(s', q', \mathbf{v}', y)$ *when the controller and adversary respectively take actions* u_C *and* u_A *being denoted as* $P_{\mathcal{Z}}\left((s', q', \mathbf{v}', y')|(s, q, \mathbf{v}, y), u_C, u_A\right)$. *The transition probability*

$$P_{\mathcal{Z}}\left((s', q', \mathbf{v}', y')|(s, q, \mathbf{v}, y), u_C, u_A\right) = \begin{cases} \sum_{\mathbf{v}''} \xi(\mathbf{v}''|\mathbf{v}) \mu_0(y', u_C|s, q, \mathbf{v}'', y) P\left((s', q', \mathbf{v}')|(s, q, \mathbf{v}), u_C, u_A\right), & \text{if } \mathcal{H}_0 \text{ holds;} \\ \mu_1(y', u_C|s, q, y) T_{\mathcal{G}}(\delta|s, u_C, u_A, s') P_{\mathcal{G}}(s'|s, u_C, u_A), & \text{if } \mathcal{H}_1 \text{ holds;} \end{cases} \tag{8.9}$$

- $Acc_{\mathcal{Z}} = Acc \times Y$ *is the set of accepting states.*

At a state $(s, q, \mathbf{v}, y)$, the information set of the controller is

$$Inf_{\mathcal{Z},C}(s, q, \mathbf{v}, y) = \{(s_0, q_0, \mathbf{v}_0, y_0), \cdots, (s, q, \bar{\mathbf{v}}, y)\}.$$

That is, the controller knows the path from the initial state of GDSG to the current state, along with the time stamps, which might have been manipulated by the adversary. The information set of the adversary is $Inf_{\mathcal{Z},A}(s, q, \mathbf{v}, y) = \{(s_0, q_0, \mathbf{v}_0, y_0), \cdots, (s, q, \mathbf{v}, y)\} \cup \{\mu\}$. That is, the adversary knows the path from the initial state to the current state, along with the correct time stamps, and the controller's policy. In the sequel, we focus on the GDSG $\mathcal{Z}$ in Definition 8.5, and denote a state $(s, q, \mathbf{v}, y)$ in $\mathcal{Z}$ as $\mathfrak{s}$. Given a run $\beta = \{(s_i, q_i, \mathbf{v}_i, y_i)\}_{i \geq 1}$ on $\mathcal{Z}$, we define $\mathsf{Untime}(\beta) = \{(s_i, q_i)\}_{i \geq 1}$ and $\mathsf{Time}(\beta) = \{(q_i, \mathbf{v}_i)\}_{i \geq 1}$, respectively. To compute the control policy that satisfies φ, we need to determine accepting runs on $\mathcal{Z}$. To this end, we introduce the concepts of generalized maximal end component (GMEC) and generalized accepting maximal end component (GAMEC) similar to Chapter 4. We note that the accepting condition for a GMEC in this chapter differs from that in Chapter 4, since we are working with timed automata.

Definition 8.6 (Sub-DSG)**.** *A sub-DSG of a DSG* $\mathcal{G} = (S, U_C, U_A, P, s_0, \Pi, \mathcal{L})$ *is a tuple* (N, D) *where* $\emptyset \neq N \subseteq S$ *is a set of states, and* $D : N \to 2^{U_C}$ *is a function such that* $D(s) \subseteq U_C(s)$ *for all* $s \in N$ *and* $\{s'|P(s'|s, u_C, u_A) > 0, \forall u_A \in U_A(s), s \in N\} \subseteq N$.

Definition 8.7. *A Generalized End Component (GEC) is a sub-DSG* (N, D) *such that the underlying directed graph* $G_{(N,D)}$ *of* (N, D) *is strongly connected. A GMEC is a GEC* (N, D)

such that there exists no other GEC $(N', D') \neq (N, D)$*, where* $N \subseteq N'$ *and* $D(s) \subseteq D'(s)$ *for all* $s \in N$*. A GAMEC is a GMEC if* $Acc \cap N \neq \emptyset$*.*

Algorithm 16 presents a procedure to compute the set of GAMECs, $\mathcal{C}$ of the GDSG $\mathcal{Z}$. Let $\mathcal{E}$ denote the set of states in a GAMEC. The correctness of Algorithm 16 is established in the following

Proposition 8.2. *Algorithm 16 returns all GAMECs of* $\mathcal{Z}$*.*

Proof. We prove the correctness of Algorithm 16 by first showing that no state or control action that belongs to a GEC will be removed. Consider a GEC (N, D).

If there exists a state $\mathfrak{s}$ and control action $u_C \in U_C(\mathfrak{s})$ such that $P(\mathfrak{s}'|\mathfrak{s}, u_C, u_A) > 0$ for all u_A and $\mathfrak{s}' \in N$, then according to lines 10 - 17 of Algorithm 16, state $\mathfrak{s}$ and control action u_C will not be removed since $D(\mathfrak{s}) \neq \emptyset$. Therefore, Algorithm 16 never removes states or actions from a GEC.

On the other hand, if there is a state $\mathfrak{s} \in N$ such that $D(\mathfrak{s}) = \emptyset$, then $\mathfrak{s}$ will be removed (lines 10 - 17 of Algorithm 16). Moreover, any state that can be steered to $\mathfrak{s}$ under some adversary action u_A will also be removed (lines 18 - 26). Thus, any state or action that does not belong to GEC will be removed by Algorithm 16, and the remaining states in (N, D) after executions from lines 10 - 26 will form the GEC.

Combining the arguments above, we have that Algorithm 16 computes a set of GECs $\{(N_i, D_i)\}_{i\geq 1}$ such that any GEC is contained by some (N_i, D_i). Then by Definition 8.7 and line 35 of Algorithm 16, we have that the result returned by Algorithm 16 is the set of GAMECs. □

The equivalence between the satisfying the MITL objective φ and reaching states in the GAMEC is stated below:

Theorem 8.1. *Given an initial state* $\mathfrak{s}_0 \in S_{\mathcal{Z}}$*, the minimum probability of satisfying* φ *is equal to the minimum probability of reaching the states* $\mathcal{E}$ *of GAMEC. That is,*

$$\min_{\tau,\xi} \mathbb{P}(\varphi|\mathfrak{s}_0) = \min_{\tau,\xi} \mathbb{P}(\textit{reach } \mathcal{E}|\mathfrak{s}_0), \tag{8.10}$$

where $\mathbb{P}(\varphi|\mathfrak{s}_0)$ *and* $\mathbb{P}(\textit{reach } \mathcal{E}|\mathfrak{s}_0)$ *are the probabilities of satisfying* φ *and reaching* $\mathcal{E}$ *when starting from* $\mathfrak{s}_0$*.*

To prove Theorem 8.1, we need an intermediate result [128].

Lemma 8.1. *Let* L *be a timed regular language. Then, a word* $\{\rho_i\}_{i\geq 1} \in \mathsf{Untime}(L)$ *if and only if there exists a sequence* $\{t_i\}_{i\geq 1}$ *such that* $t_i \in \mathbb{Q}$ *and the timed word* $\{\rho_i, t_i\}_{i\geq 1} \in L$*.*

Lemma 8.1 indicates that we can analyze a timed word by focusing on its untimed projection and the corresponding time sequence. We use this to prove Theorem 8.1.

Algorithm 16 Computing the set of GAMECs $\mathcal{C}$.

Input: GDSG $\mathcal{Z}$
Output: Set of GAMECs $\mathcal{C}$
Initialization: Let $D(\mathfrak{s}) \leftarrow U_C(\mathfrak{s})$ for all $\mathfrak{s} \in S$. Let $\mathcal{C} \leftarrow \emptyset$ and $\mathcal{C}_{temp} \leftarrow \{S\}$
repeat
    $\mathcal{C} \leftarrow \mathcal{C}_{temp}$, $\mathcal{C}_{temp} \leftarrow \emptyset$
    for $N \in \mathcal{C}$ **do**
        $R \leftarrow \emptyset$
        Let $SCC_1, \cdots, SCC_n$ be the set of strongly connected components of underlying digraph $G_{(N,D)}$
        for $i = 1, \cdots, n$ **do**
            for each state $\mathfrak{s} \in SCC_i$ **do**
                $D(\mathfrak{s}) \leftarrow \{u_C \in U_C(\mathfrak{s}) | \mathfrak{s}' \in N, P(\mathfrak{s}'|\mathfrak{s}, u_C, u_A) > 0, \ \forall u_A \in U_A(\mathfrak{s})\}$
                if $D(s) = \emptyset$ **then**
                    $R \leftarrow R \cup \{\mathfrak{s}\}$
                end if
            end for
        end for
        while $R \neq \emptyset$ **do**
            dequeue $\mathfrak{s} \in R$ from R and N
            if $\exists \mathfrak{s}' \in N$ and $u_C \in U_C(\mathfrak{s}')$ such that $P(\mathfrak{s}|\mathfrak{s}', u_C, u_A) > 0$ for some $u_A \in U_A(\mathfrak{s}')$ **then**
                $D(\mathfrak{s}') \leftarrow D(\mathfrak{s}') \setminus \{u_C\}$
                if $D(\mathfrak{s}') = \emptyset$ **then**
                    $R \leftarrow R \cup \{\mathfrak{s}'\}$
                end if
            end if
        end while
        for $i = 1, \cdots, n$ **do**
            if $N \cap SCC_i \neq \emptyset$ **then**
                $\mathcal{C} \leftarrow \mathcal{C}_{temp} \cup \{N \cap SCC_i\}$
            end if
        end for
    end for
until $\mathcal{C} = \mathcal{C}_{temp}$
for $N \in \mathcal{C}$ **do**
    if $Acc_{\mathcal{Z}} \cap N = \emptyset$ **then**
        $\mathcal{C} = \mathcal{C} \setminus N$
    end if
end for
return $\mathcal{C}$

Proof of Theorem 8.1. We establish that satisfying φ is equivalent to reaching the set of states $\mathcal{E}$. Then, we need to show that any accepting run will reach $\mathcal{E}$, and any run that reaches $\mathcal{E}$ is accepting. Let $\mathcal{L}$ denote the timed language accepted by $\mathcal{Z}$. Let $\mathsf{Untime}(\mathcal{L})$ be the language obtained from $\mathcal{L}$ by discarding the clock valuation and internal state components.

First, we prove that any accepting run β of GDSG $\mathcal{Z}$ reaches $\mathcal{E}$. We use a contradiction argument. Suppose there exists an accepting run β that does not reach $\mathcal{E}$. Since β satisfies φ, we must have that β contains some accepting state in $Acc_{\mathcal{Z}}$ infinitely many times (Definition 3.9). This implies that there exists a GEC that contains some state $\mathfrak{s} \in Acc_{\mathcal{Z}}$ and $\mathfrak{s} \notin \mathcal{E}$, which violates Proposition 8.2.

Next, we show that any run β that reaches $\mathcal{E}$ is accepting. We use Lemma 8.1. Since GAMECs are strongly connected and each GAMEC contains at least one accepting state, reaching $\mathcal{E}$ is equivalent to reaching some accepting state infinitely often, which agrees with the acceptance condition of $\mathsf{Untime}(\mathcal{L})$. Hence, we have $\mathsf{Untime}(\beta) \in \mathsf{Untime}(\mathcal{L})$. Now, from Eqn. (8.3), we have that a transition in $\mathcal{P}$, and therefore in $\mathcal{Z}$, exists if and only if no clock constraint is violated, i.e., $(q, \mathbf{v}) \xrightarrow{L(s'),\delta} (q', \mathbf{v}')$. Otherwise, the transition probability is 0, and hence the run β does not exist, which establishes the claim. Given that $\mathsf{Untime}(\beta) \in \mathsf{Untime}(L)$ holds, and $\mathsf{Time}(\beta)$ never violates the clock constraints defined by TBA $\mathcal{A}$ for any run that reaches $\mathcal{E}$, we have that the set of runs that reach $\mathcal{E}$ is in language $\mathcal{L}$ by Lemma 8.1.

Combining the two arguments above, we observe that satisfying φ is equivalent to reaching the set $\mathcal{E}$. This gives $\min_{\tau,\xi} \mathbb{P}(\varphi|\mathfrak{s}_0) = \min_{\tau,\xi} \mathbb{P}(\text{reach } \mathcal{E}|\mathfrak{s}_0)$, completing the proof. □

Let the vector $\mathbf{Q}(\mathfrak{s}) \in \mathbb{R}^{|S_{\mathcal{Z}}|}$ represent the probability of satisfying φ when starting from a state $\mathfrak{s} = (s, q, \mathbf{v}, y)$ in $\mathcal{Z}$.

Proposition 8.3. *Let* $\mathbf{Q} := \max_{\mu} \min_{\tau,\xi} \mathbb{P}(\varphi)$ *be the probability of satisfying* φ*. Then,*

$$\mathbf{Q}((s,q,\mathbf{v},y)) = \max_{\mu} \min_{\tau,\xi} \sum_{u_C \in U_C} \sum_{u_A \in U_A} \sum_{(s',q',\mathbf{v}',y) \in S_{\mathcal{Z}}} \tau((s,q,\mathbf{v},y),u_A)\mathbf{Q}((s',q',\mathbf{v}',y')) \cdot P_{\mathcal{Z}}\left((s',q',\mathbf{v}',y')|(s,q,\mathbf{v},y),u_C,u_A\right), \ \forall (s,q,\mathbf{v},y). \quad (8.11)$$

Moreover, the value vector is unique.

We define the following operators to prove Proposition 8.3.

$$(M_\mu \mathbf{Q})(\mathfrak{s}) = \min_{\tau,\xi} \sum_{\mathfrak{s}'} P(\mathfrak{s}'|\mathfrak{s},\mu,(\tau,\xi))\mathbf{Q}(\mathfrak{s}'),$$

$$(M\mathbf{Q})(\mathfrak{s}) = \max_{\mu} \min_{\tau,\xi} \sum_{\mathfrak{s}'} P(\mathfrak{s}'|\mathfrak{s},\mu,(\tau,\xi))\mathbf{Q}(\mathfrak{s}'),$$

where $P(\mathfrak{s}'|\mathfrak{s},\mu,(\tau,\xi))$ is the probability of transiting from state $\mathfrak{s}$ to $\mathfrak{s}'$, given policies μ and

τ. The operators M_μ and M are characterized in the following lemma. The proof of the lemma and Proposition 8.3 can be found in the Appendix.

Lemma 8.2. *The sequence of value vectors obtained by composing operators M_μ and M is convergent.*

Proof. Given a control policy μ, the GDSG $\mathcal{Z}$ is reduced to an MDP, $\mathcal{M}$. Then, the composition of M_μ corresponds to a value iteration on $\mathcal{M}$. The convergence of M_μ can be shown following the approach in [101].

Next, we show that the sequence obtained by composing M is bounded and monotone. We observe that $M\mathbf{Q}(\mathfrak{s})$ is a convex combinations of all the neighboring states of $\mathfrak{s}$. Moreover, $\mathbf{Q}(\mathfrak{s}) \in [0,1]$ for all $\mathfrak{s}$, and is therefore bounded. We show that the sequence of value vectors is monotonically non-decreasing by induction. Define $M^{-1}\mathbf{Q} := \mathbf{0}$, and $M^0\mathbf{Q}(\mathfrak{s}) = 0$ for $\mathfrak{s} \notin \mathcal{E}$, and $M^0\mathbf{Q}(\mathfrak{s}) = 1$ for $\mathfrak{s} \in \mathcal{E}$. Then, $M^{-1}\mathbf{Q} \leq M^0\mathbf{Q}$. Suppose the sequence of value vectors is monotonically non-decreasing up to iteration k. We have

$$
\begin{aligned}
& M^{k+1}\mathbf{Q}(\mathfrak{s}) \\
\geq & \min_{\tau,\xi} \Bigg\{ \sum_{u_C \in U_C} \sum_{u_A \in U_A} \sum_{(s',q',\mathbf{v}',y) \in S_{\mathcal{Z}}} \tau((s,q,\mathbf{v},y),u_A)\mathbf{Q}((s',q',\mathbf{v}',y')) \\
& \qquad \cdot P_{\mathcal{Z}}\left((s',q',\mathbf{v}',y')|(s,q,\mathbf{v},y),u_C,u_A\right) \Bigg\} && (8.12) \\
\geq & \min_{\tau,\xi} \Bigg\{ \sum_{u_C \in U_C} \sum_{u_A \in U_A} \sum_{(s',q',\mathbf{v}',y) \in S_{\mathcal{Z}}} \tau((s,q,\mathbf{v},y),u_A)\mathbf{Q}((s',q',\mathbf{v}',y')) \\
& \qquad \cdot P_{\mathcal{Z}}^{k-1}\left((s',q',\mathbf{v}',y')|(s,q,\mathbf{v},y),u_C,u_A\right) \Bigg\} && (8.13) \\
= & M^k\mathbf{Q}(\mathfrak{s}), && (8.14)
\end{aligned}
$$

where $P_{\mathcal{Z}}^{k-1}\left((s',q',\mathbf{v}',y')|(s,q,\mathbf{v},y),u_C,u_A\right)$ is obtained by substituting $\mu^{k-1}(g',u_C|g,\mathbf{v})$ into (8.9), inequality (8.12) holds since $M^{k+1}\mathbf{Q}$ corresponds to a maximizing policy μ^{k+1}, (8.13) holds by induction, and (8.14) follows from the construction of μ^k. Therefore, $\mathbf{Q}^{k+1} \geq \mathbf{Q}^k$, implying the sequence of value vectors is monotonically non-decreasing. From the boundedness and monotonocity of $M\mathbf{Q}$, the sequence is a Cauchy sequence that converges to a value $\mathbf{Q}^*$. □

These results enable determining an optimal control policy using a value-iteration based algorithm. Algorithm 17 computes the value vector at each iteration. The value vector is updated following Proposition 8.3. Given the optimal value vector $\mathbf{Q}^*$ and the Stackelberg setting, we can extract the optimal controller's policy as the maximizer of $\mathbf{Q}^*$ by solving a linear program. The convergence of Algorithm 17 is discussed in the following theorem.

Algorithm 17 Computing an optimal control policy.

1: **Input**: GDSG $\mathcal{Z}$
2: **Output:** value vector $\mathbf{Q}$
3: **Initialization:** $\mathbf{Q}^0 \leftarrow \mathbf{0}$, $\mathbf{Q}^1(\mathfrak{s}) \leftarrow 1$ for $\mathfrak{s} \in Acc_{\mathcal{Z}}$, $\mathbf{Q}^1(\mathfrak{s}) \leftarrow 0$ otherwise, $k \leftarrow 0$
4: **while** $\max\{|\mathbf{Q}^{k+1}(\mathfrak{s}) - \mathbf{Q}^k(\mathfrak{s})| : \mathfrak{s} \in S\} > \epsilon$ **do**
5: $k \leftarrow k+1$
6: **for** $\mathfrak{s} \notin Acc_{\mathcal{Z}}$ **do**
7: $\mathbf{Q}^{k+1}(\mathfrak{s}) \leftarrow \max_{\mu} \min_{\tau,\xi} \left\{ \sum_{u_C \in U_C} \sum_{u_A \in U_A} \sum_{(s',q',\mathbf{v}',y) \in S_{\mathcal{Z}}} \tau((s,q,\mathbf{v},y),u_A)\mathbf{Q}((s',q',\mathbf{v}',y'))P_{\mathcal{Z}}\left((s',q',\mathbf{v}',y')|(s,q,\mathbf{v},y),u_C,u_A\right) \right\}$
8: **end for**
9: **end while**
10: **return** $\mathbf{Q}^k$

Theorem 8.2. *Algorithm 17 converges in a finite number of iterations. Moreover, the value vector returned by Algorithm 17 is in an ϵ-neighborhood of $\mathbf{Q}^*$.*

Proof of Theorem 8.2. We prove convergence by by showing that the sequence of value vectors computed in Algorithm 17 is bounded and monotonically non-decreasing. Line 4 of Algorithm 17 serves as our induction base, i.e., $\mathbf{Q}^1 \geq \mathbf{Q}^0$. Line 8 of Algorithm 17 is equivalent to computing $\mathbf{Q}^{k+1}$ as $\mathbf{Q}^{k+1} = M\mathbf{Q}^k$. From Lemma 8.2, $\mathbf{Q}^{k+1} \geq \mathbf{Q}^k$. Convergence follows from the Monotone Convergence theorem [119]. That the control policy is within an ϵ-neighborhood of SE follows from Line 5 of Algorithm 17. □

8.4 Case Study

This section evaluates the proposed solution approach on a signalized traffic network and a two-tank system. The simulations were carried out using MATLAB on a Macbook Pro with a 2.6GHz Intel Core i5 CPU and 8GB RAM. The Appendix contains an example on two-tank system.

8.4.1 Signalized Traffic Network Model

We consider signalized traffic network under the remote control of a transportation management center (TMC). A signalized traffic network consists of a set of links $\{1, 2, \ldots, L\}$ and intersections $\{1, 2, \ldots, N\}$ [129]. Each intersection can take a 'red' signal which will not allow vehicles to pass through the intersection, or a 'green' signal which will allow vehicles to pass. The number of vehicles in link l at a time k is $x_l(k)$ and $\bar{x}_l$ denotes the capacity of link l.

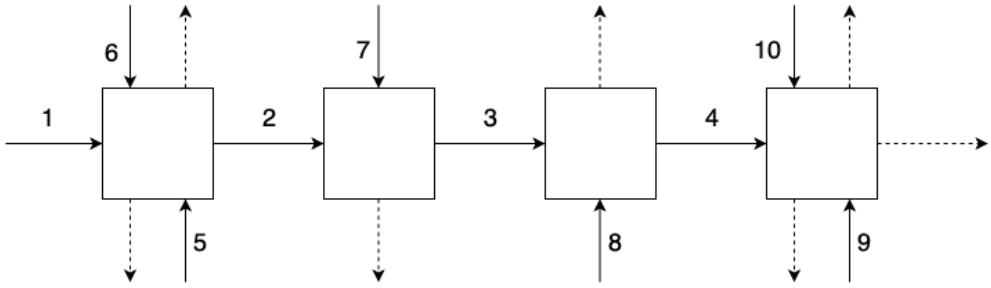


Figure 8-1: Representation of a *signalized traffic network*. The network consists of 4 intersections and 16 links. Intersections are represented by squares, and links by arrows. Dotted arrows denote outgoing links that are not explicitly modeled.

The number of vehicles entering the traffic network at a time k is assumed to follow a Poisson distribution. Vehicles can travel through a link if and only if the subsequent intersection in the direction of travel is green. The link is then said to be *actuated*. We assume that the flow rate c_l of each link l is given and fixed. The TMC is given an MITL objective that needs to be satisfied on the network. When the TMC issues a green signal at an intersection n, the turn ratio $\gamma_{ll'} \in [0, 1]$ denotes the fraction of vehicles that will move to link l' from link l through intersection n. The maximum number of vehicles that can be routed to l' from l is determined by the supply ratio $\alpha_{ll'}$ of link l', which is determined by the remaining capacity $\bar{x}_{l'} - x_{l'}(k)$ of link l'. Given the above parameters, the dynamics of the link queues can be determined [129].

We assume there is an adversary who can initiate actuator and timing attacks. An actuator attack will tamper with the traffic signal issued by the TMC. For instance, if the TMC actuates a link l at time k and the adversary attacks link l, then this link will not be actuated at time k. A timing attack will manipulate the timing information perceived by the TMC. Hence, any time stamped measurement $\{x_l, k\}$ received by the TMC indicating the number of vehicles at link l at time k might be manipulated to $\{x_l, k'\}$, where k' is the time stamp that has been changed by the adversary.

The signalized traffic network model can be mapped to a DSG in the following way. States of the DSG are obtained by partitioning the number of vehicles on each link (e.g., box partition) [129]. The control action set at each intersection models which links can be actuated. The action set is then realized by taking the Cartesian product of the action sets at each intersection. The realized traffic signal at an intersection is jointly determined by the actions of the TMC and adversary. The transition and duration probability distributions between states are obtained from Algorithm 15.

A representation of the signalized traffic network is shown in Fig. 8-1. The network consists of 4 intersections (squares) and 16 links (arrows). We denote the intersections

with incoming links 1, 2, 3, and 4 as intersections 1, 2, 3, and 4, respectively. The links represented by dotted arrows are not explicitly modeled [129]. For each intersection in Fig. 8-1, the links that can be actuated by the TMC are given as follows:

- Intersection 1: $\{\{1\},\{5,6\}\}$;
- Intersection 2: $\{\{2\},\{7\}\}$;
- Intersection 3: $\{\{3\},\{8\}\}$;
- Intersection 4: $\{\{4\},\{9,10\}\}$.

We assume that the TMC can actuate exactly one subset of links at each intersection so that no safety constraint will be violated. The link capacities are set to $\bar{x}_1 = \cdots = \bar{x}_5 = 30$ and $\bar{x}_6 = \cdots = \bar{x}_{10} = 40$. Flow rates associated to each link are set to $c_1 = \cdots = c_4 = 10$, $c_5 = \cdots = c_{10} = 5$ [129]. The supply ratios $\alpha_{ll'} = 1$ for all l, l', and the turn ratios are set to $\gamma_{12} = 0.3$, $\gamma_{23} = \gamma_{34} = \gamma_{52} = \gamma_{62} = \gamma_{73} = \gamma_{84} = 0.5$. Vehicles entering a link in $(l_1, \ldots, l_{10})$ follow a Poisson distribution with mean $(5, 0, 0, 0, 5, 5, 0, 0, 5, 5)$. We consider a time horizon of length 5. The controller's strategy to detect a timing attack is to compare the deviation between its estimated and observed clock valuations with a pre-specified threshold $\mathbf{e} = 2$. In particular, when $\|\boldsymbol{\lambda} - \mathbf{v}\| \leq 2$, hypothesis $\mathcal{H}_0$ holds and no timing attack is detected by the controller. When $\|\boldsymbol{\lambda} - \mathbf{v}\| > 2$, hypothesis $\mathcal{H}_1$ holds and an alarm indicating a timing attack is triggered. In this case, the FSC equipped by the controller has 5 internal states.

	Intersection			
Time	1	2	3	4
1	G	R	G	R
2	R	G	R	G
3	G	G	R	G
4	G	G	G	G
5	G	R	G	G

Table 8.1: A sample sequence of the traffic light realized at each intersection for the MITL specification $\varphi_3 = \diamondsuit_{[0,5]}((x_2 \leq 10) \wedge (x_3 \leq 10) \wedge (x_4 \leq 10))$. The letter 'R' represents a 'red' signal, and 'G' represents 'green' signal.

The TMC is given one of the following MITL objectives.

1. The number of vehicles at link 2 is eventually below 10 before deadline $d = 5$: $\varphi_1 = \diamondsuit_{[0,5]}(x_2 \leq 10)$.
2. The number of vehicles at link 2 and 3 are eventually below 10 before deadline $d = 5$: $\varphi_2 = \diamondsuit_{[0,5]}((x_2 \leq 10) \wedge (x_3 \leq 10))$.

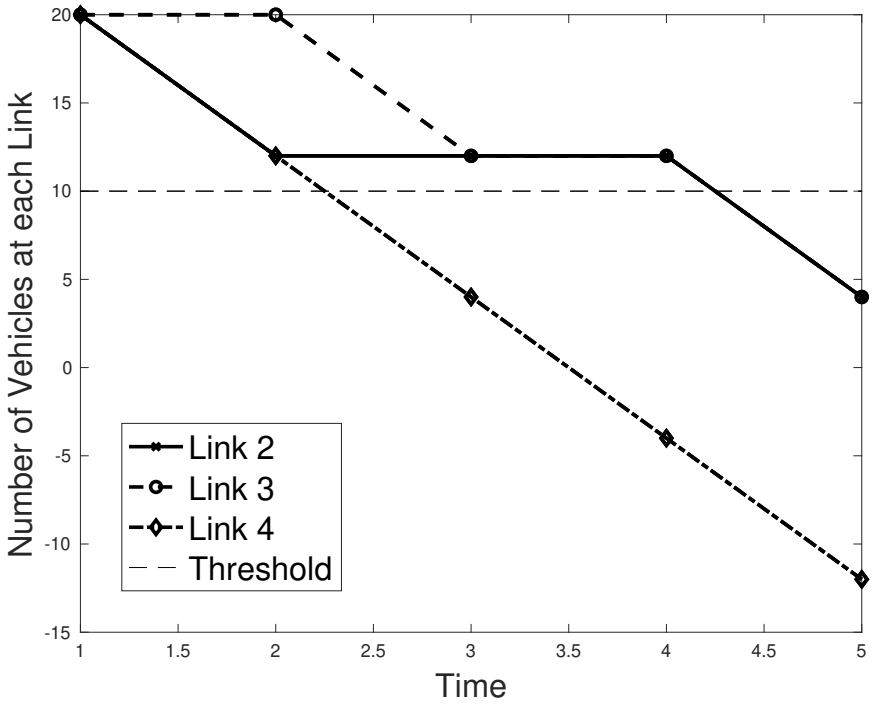


Figure 8-2: Number of vehicles on Links 2, 3, and 4 at each time corresponding to the MITL formula $\varphi_3 = \diamondsuit_{[0,5]} ((x_2 \leq 10) \wedge (x_3 \leq 10) \wedge (x_4 \leq 10))$. In the presence of an adversary, the controller adopts an FSC-based policy with one realization shown in Table 8.1. The dotted horizontal line is the threshold for the maximum number of vehicles allowed (= 10). The three curves indicate that the number of vehicles in the links satisfies the MITL objective since they are each lower than 10 before 5 time units.

3. The number of vehicles at link 2, 3 and 4 are eventually below 10 before deadline $d = 5$:
   $\varphi_3 = \Diamond_{[0,5]} ((x_2 \leq 10) \wedge (x_3 \leq 10) \wedge (x_4 \leq 10))$.

Our experiments yield the probability of satisfying each specification as: $\mathbb{P}(\varphi_{\mathbf{1}}) = \mathbf{0.723}$, $\mathbb{P}(\varphi_{\mathbf{2}}) = \mathbf{0.371}$, and $\mathbb{P}(\varphi_{\mathbf{3}}) = \mathbf{0.333}$. These values agree with intuition since φ_3 being satisfied implies φ_2 holds true, which in turn implies that φ_1 is true.

We compare our approach for the objective φ_3 with two baselines. In the first baseline, the TMC issues periodic green signals for links $1, 2, 3$, and 4 at intersections $1, 2, 3$, and 4, respectively. In the second baseline, the TMC always issues green signals for links $1, 2, 3, 4$ at intersections $1, 2, 3, 4$.

For the two baseline scenarios, the TMC commits to deterministic strategies. The adversary's actuator attack strategies are as follows. In the first case, the adversary launches actuator attacks when the TMC issues a green signal, and does not attack when the TMC issues a red signal. As a result, the realized traffic signal will be red for all time at every intersection. In the second case, the adversary launches an actuator attack at every time instant. This results in the realized traffic signal being red for all time at each intersection. As a consequence, the number of vehicles in links $2, 3$ and 4 will reach their capacities and the links will be congested for the rest of the time horizon. Therefore, the probabilities of satisfying the MITL specification using the baselines are zero.

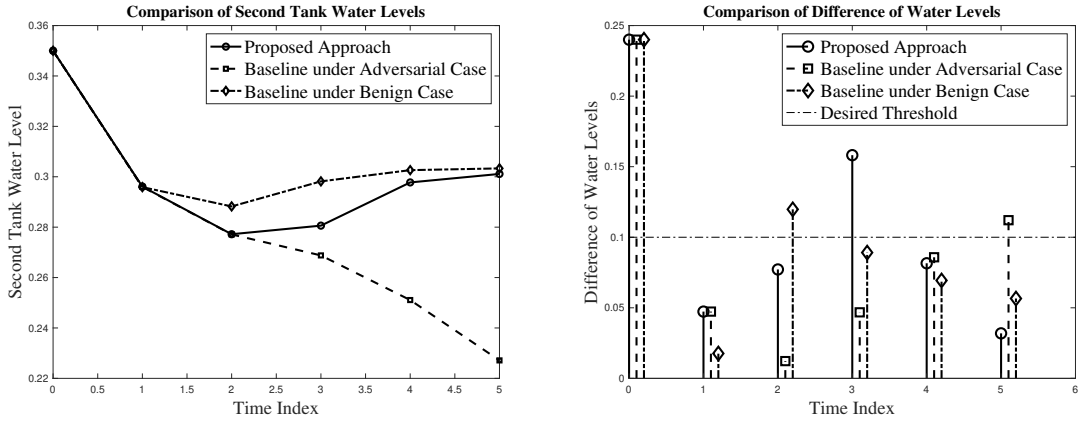
Table 8.1 shows a realization of the traffic signals when the controller adopts an FSC-based policy proposed in Section 8.3. Fig. 8-2 shows the number of vehicles in each link for this realization. The graph indicates that the controller's policy is successful in ensuring that the MITL objective is satisfied. Moreover, if the adversary's timing strategy is such that when the difference in the manipulated and actual clock valuations is less than 2 (the pre-specified threshold), it remains stealthy, even though this is not an explicitly specified goal.

The construction of the DSG using Algorithm 15 takes 24.22 seconds. The computation of the global DSG takes 367.9 seconds. Given the global DSG, Algorithm 17 takes 644.8 seconds to compute the controller's FSC.

8.4.2 Two-Tank System

We demonstrate our solution approach with simulations carried out on the control of a two-tank system [130]. The system is described by $x(k+1) = Ax(k) + Bu(k) + w(k)$, where $x(k) = [x_1(k), x_2(k)]^T$, $u(k)$, and $w(k)$ are state variables representing water levels, control input representing the inflow rate, and stochastic disturbance at time k, respectively. The controller transmits a control signal $u_C(k)$ to the actuator through a wireless communication channel. We set the initial levels in the two tanks to $x(0) = [0.11, 0.35]^T$.

The system is subject to an attack initiated by an intelligent adversary. The control signal $u(k)$ received by the actuator is compromised as $u(k) = u_C(k) + u_A(k)$ due to the actuator attack, where $u_C(k)$ and $u_A(k)$ correspond to signals sent by the controller and



(a) (b)

Figure 8-3: Evaluation on a two-tank system for an MITL specification that requires water levels in the tanks to be at least 0.3, and to be within 0.1 of each other, before time $k = 5$. An FSC-based controller policy is compared with a baseline policy that does not account for the presence of an adversary. Fig. 8-3a shows the water level in the second tank, using the two policies. The solid line represents the FSC-based policy, while the dashed and dash-dot lines represent the baseline in the presence and absence of the adversary, respectively. The absolute value of the difference between water levels in the two tanks using the two policies is presented in Fig. 8-3b. The solid line with circle markers represents the FSC-based policy, while the dashed line and dash-dot line represent the baseline policy under adversarial and benign environments, respectively. We observe that the baseline policy satisfies the MITL objective in the absence of the adversary, but fails to do so when an adversary is present. The FSC-based policy, in contrast, satisfies the MITL objective in the presence of the adversary.

adversary [49]. Due to the timing attack, the time-stamped measurement $\{x, k\}$ indicating the water level at time k is manipulated as $\{x, k'\}$, where k' is the time stamp that has been modified by the adversary.

The state space (water levels in tanks) is partitioned into 49 rectangular regions, i.e., the water level in each tank is divided into 7 discrete intervals with discretization resolution 0.1, each representing a state of the DSG. The control and adversary signals are in the ranges $[0, 5 \times 10^{-4}]$ and $[0, 2 \times 10^{-4}]$, respectively [130]. Control and adversary action sets are obtained by discretization of these sets of inputs. The disturbance $w(k)$ is zero mean i.i.d. Gaussian with covariance 1.5×10^{-5}. The transition and duration probabilities are obtained by Algorithm 15. This procedure took about 18 seconds.

The system needs to satisfy an MITL specification given by

$$\varphi = \Diamond_{[0,5]} \left(\bigvee_{z} \bigwedge_{i \in \{1,2\}} (z \leq x_i \leq z + 0.1) \right),$$

where $z \in \{0.3, 0.4, 0.5, 0.6\}$. That is, before time $k = 5$, the water levels in the two tanks should lie in the same discretization interval and are each required to be no less than 0.3. If the MITL specification is satisfied, the difference between water levels in the two tanks should be at most 0.1.

We compare our FSC-based policy with a baseline. The baseline does not account for the presence of the adversary. The results of our experiments are presented in Fig. 3. The baseline is evaluated for scenarios where the adversary is present and the adversary is absent. When there is no adversary, we observe that the baseline policy satisfies the MITL objective (the water levels in the tanks are 0.35 and 0.30, and the difference in the levels is 0.05). However, when this policy is used in the presence of the adversary, we observe that the water level in the second tank falls below 0.3, and the difference in the levels exceeds 0.1, thereby violating the specification. This necessitates the use of an alternative control strategy for systems under attacks. Using our approach, we observe that the water levels in the two tanks are 0.33 and 0.30, and the difference in the levels is 0.03. Moreover, these water levels are attained before the required deadline of $k = 5$, which satisfies the MITL objective.

8.5 Conclusion

In this chapter, we investigated the problem of synthesizing controllers for time critical CPSs under attack. We proposed durational stochastic games to capture the interaction between the controller and adversary, and also account for time taken for transitions between states. The CPS had to satisfy a time-dependent objective specified as an MITL formula. We used a timed automaton representation of the MITL formula, the DSG, and a representation of the controller policy as a finite state controller to synthesize controller policies that would satisfy the MITL objective under actuator and timing attacks carried out by the adversary. We evaluated our solution method on a representation of a signalized traffic network and a two-tank system.

Chapter 9

Abstraction-Free Control Synthesis under Temporal Logic Constraints

9.1 Introduction

Chapter 4 to 8 rely on the construction of a finite abstract model such as stochastic game and durational stochastic game to capture the interaction between the CPS and adversary. Control synthesis based on finite abstract models are categorized as *abstraction-based* approaches. These approaches are computationally demanding and suffer from the curse of dimensionality. To avoid the computation required for constructing the finite abstractions, researchers have investigated control synthesis on the continuous state space when the system is operated under benign environment. Techniques include formulating the LTL constraint as a mixed integer linear program [38], a sequence of stochastic reachability problems [39], and a mixed continuous-discrete HJB equation [40]. Solving for the controller thus requires numerical methods which can be computationally expensive.

In this chapter, we consider control synthesis for CPS under LTL constraints without computing a finite abstraction, assuming the CPS are operated in the *absence* of an adversary. We aim to compute a feedback controller such that a control affine system satisfies a given LTL specification from the fragment of LTL without next operator. We present a control barrier function (CBF) based framework to compute the controller. There are two main advantages of our approach compared to the state of the art. First, we avoid both finite-state abstraction of the CPS and approximate solution of the HJB equation, and thus reduce the computational complexity. Second, we introduce time-varying *guard functions* that render our approach feasible for a broad class of LTL properties. This chapter makes the following specific contributions:

- We present a CBF-based approach to synthesize a controller for CPS under LTL constraints. The proposed approach is provably correct and avoids explicit construction of finite abstractions for CPS.

- We construct a sequence of formulas that correspond to an accepting trace of the CPS, and develop a methodology to construct CBFs for each formula. When designing the CBF, we introduce the concept of guard function, which implicitly encodes the time that each formula needs to be satisfied and enhances the feasibility of the CBF constraints.

- We compute the set of controllers that satisfy each constructed formula using two types of time varying CBFs. We show that by satisfying the CBF constraints for all formulas, the LTL specification is satisfied.

- A numerical case study on robotic motion planning is presented as evaluation. The proposed approach successfully synthesizes a controller for a specification that is infeasible under the state of art.

The remainder of this chapter is organized as follows. Section 9.2 introduces background on control barrier functions. Section 9.3 presents the system model and problem formulation on abstraction-free control synthesis. Section 9.4 presents our proposed solution approach. Section 9.5 evaluates our proposed approach on multi-agent system, and Section 9.6 concludes this chapter.

9.2 Preliminary Background

A function $f : \mathbb{R}^n \times [0, \infty) \mapsto \mathbb{R}$ is Lipschitz continuous on $\mathcal{X} \subset \mathbb{R}^n$ if there exists some constant $K > 0$ such that $\|f(x_1) - f(x_2)\| \leq K\|x_1 - x_2\|$ for all $x_1, x_2 \in \mathcal{X}$, where $\|\cdot\|$ is the Euclidean norm. A function f is locally Lipschitz continuous if there exist constants $\tau > 0$ and $K > 0$ such that $\|f(x_1) - f(x_2)\| \leq K\|x_1 - x_2\|$ for all $\|x_1 - x_2\| \leq \tau$.

A continuous function $\alpha : [0, a) \mapsto [0, \infty)$ belongs to class $\mathcal{K}$ if it is strictly increasing and $\alpha(0) = 0$. A continuous function $\alpha : [-b, a) \mapsto (-\infty, \infty)$ is said to belong to extended class $\mathcal{K}$ if it is strictly increasing and $\alpha(0) = 0$ for some $a, b > 0$. The following lemma gives smooth approximations of min and max operators.

Lemma 9.1 (Approximation of min and max Operators [131])**.** *Consider a set of functions $h_i(x, t)$. Then for $\lambda > 0$*

$$\min_i h_i(x,t) \geq -\ln\left(\sum_{i=1}^{k} \exp\left(-h_i(x,t)\right)\right),$$
$$\max_i h_i(x,t) \geq \frac{\sum_i h_i(x,t)\exp(\lambda h_i(x,t))}{\sum_i \exp(\lambda h_i(x,t))}.$$

Consider a continuous-time control-affine system

$$\dot{x} = f(x) + g(x)u \tag{9.1}$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^n$ is the system state and $u \in \mathcal{U} \subseteq \mathbb{R}^m$ is input provided by the controller. Vector-valued and matrix-valued functions $f(x)$ and $g(x)$ are of appropriate dimensions and are Lipschitz continuous.

Let a safety set $\mathcal{C}$ be defined as

$$\mathcal{C} = \{x \in \mathcal{X} : h(x) \geq 0\}, \tag{9.2}$$

where $h : \mathcal{X} \mapsto \mathbb{R}$ is a continuously differentiable function. We say system (9.1) is safe with respect to $\mathcal{C}$ if $x \in \mathcal{C}$ for all time $t \geq 0$.

Control barrier function (CBF)-based approaches have been used to guarantee safety of system (9.1) with respect to safe set $\mathcal{C}$. We give the definition of zeroing CBF as follows.

Definition 9.1 (Zeroing CBF (ZCBF) [44])**.** *Consider a dynamical system* (9.1) *and a continuously differentiable function $h : \mathcal{X} \mapsto \mathbb{R}$. If there exists a locally Lipschitz extended class $\mathcal{K}$ function α such that for all $x \in \mathcal{X}$ the following inequality holds*

$$\sup_{u \in \mathcal{U}} \left\{ \frac{\partial h(x)}{\partial x} f(x) + \frac{\partial h(x)}{\partial x} g(x) u + \alpha(h(x)) \right\} \geq 0,$$

then function h is a ZCBF.

A sufficient condition for the safety guarantee can be derived using ZCBF as follows.

Lemma 9.2 ([44])**.** *Given a dynamical system* (9.1) *and a safety set* (9.2) *defined by some continuously differentiable function $h : \mathcal{X} \mapsto \mathbb{R}$, if h is a ZCBF defined on $\mathcal{X}$, then $\mathcal{C}$ is forward invariant.*

Using Lemma 9.2, one can solve for the controller guaranteeing the safety of system (9.1) at each time using a quadratic program [44]:

$$\min_{u} \; u^\top R(x) u + Q(x)^\top u \tag{9.3a}$$

$$\text{s.t. } \frac{\partial h(x)}{\partial x} f(x) + \frac{\partial h(x)}{\partial x} g(x) u + \alpha(h(x)) \geq 0 \tag{9.3b}$$

$$u \in \mathcal{U} \tag{9.3c}$$

where $R(x) \in \mathbb{R}^m$ is positive definite. Finite time convergence CBF (FCBF) has been proposed to characterize the convergence time of system (9.1) to $\mathcal{C}$. We introduce FCBF as follows.

Definition 9.2 (Finite Time Convergence CBF (FCBF))**.** *Consider a dynamical system* (9.1) *and a continuously differentiable function $h : \mathcal{X} \mapsto \mathbb{R}$. If there exist constants $\rho \in [0, 1)$ and $\gamma > 0$ such that for all $x \in \mathcal{X}$ the following inequality holds*

$$\sup_{u \in \mathcal{U}} \left\{ \frac{\partial h(x)}{\partial x} f(x) + \frac{\partial h(x)}{\partial x} g(x) u + \gamma \cdot sgn(h(x,t)) |h(x)|^\rho \right\} \geq 0, \tag{9.4}$$

then function h is a FCBF.

The set of control inputs satisfying Eqn. (9.4) provides the following guarantee.

Lemma 9.3 ([132]). *Consider a dynamical system* (9.1) *and a set $\mathcal{C} = \{x : h(x) \geq 0\}$. If h is an FCBF defined on $\mathcal{X}$, then the control signals satisfying Eqn.* (9.4) *guarantees that there exists some finite $T \in \left[0, \frac{|h(x(0))|^{1-\rho}}{\gamma(1-\rho)}\right]$ such that $x(T) \in \mathcal{C}$ for any initial state $x(0) \in \mathcal{X}$. Moreover, the system state $x \in \mathcal{C}$ for all time $t' \geq T$.*

9.3 System Model and Problem Formulation

In this section, we present the system model and problem formulation. Consider system (9.1) with f and g being locally Lipschitz continuous. Given the current system state x, a feedback controller is a function $\mu : \mathcal{X} \times [0, \infty) \mapsto \mathcal{U}$.

The system is given a specification φ modeled using LTL without next operator, denoted as $\text{LTL}_{\setminus\bigcirc}$. The class of $\text{LTL}_{\setminus\bigcirc}$ formulas are inductive defined as

$$\varphi = True \mid \pi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \text{U} \varphi_2.$$

We note that this differs with the definition in Chapter 3 in the sense that the until operator **U** is not considered. Thus, $\text{LTL}_{\setminus\bigcirc}$ formulas can also be equivalently expressed using DRAs.

Let Π be a finite set of atomic propositions. We define a labeling function $L : \mathcal{X} \mapsto 2^{\Pi}$ that maps any state $x \in \mathcal{X}$ to a subset of atomic propositions that hold true at x. We also define $[\![\pi]\!] = \{x | \pi \in L(x)\}$ to be the set of states that satisfies the atomic proposition $\pi \in \Pi$. In this chapter, we assume that $[\![\pi]\!]$ is a closed set for all $\pi \in \Pi$, and $[\![\pi]\!]$ can be represented as $[\![\pi]\!] = \{x | Z_\pi(x) \geq 0\}$, where $Z_\pi : \mathbb{R}^n \mapsto \mathbb{R}$ is a bounded and continuously differentiable function. We slightly overload the notation $[\![\cdot]\!]$, and define the states that satisfy a subset of atomic propositions $P \in 2^{\Pi}$ as

$$[\![P]\!] = \begin{cases} \mathcal{X} \setminus \cup_{\pi \in \Pi} [\![\pi]\!] & \text{if } P = \emptyset \\ \cap_{\pi \in P} [\![\pi]\!] \setminus \cup_{\pi \in \Pi \setminus P} [\![\pi]\!] & \text{otherwise} \end{cases} \tag{9.5}$$

That is, $[\![P]\!]$ is the subset of system states $\mathcal{X}$ that satisfy all and only propositions in P [133].

We define the trajectory of system (9.1) as $\mathbf{x} : [0, \infty) \mapsto \mathcal{X}$ that maps from any time $t \geq 0$ to the system state $x(t)$. We then define the trace of a trajectory $\mathbf{x}$ as follows.

Definition 9.3 (Trace of Trajectory [133]). *An infinite sequence $Trace(\mathbf{x}) = P_0, P_1, \ldots$, where $P_i \in 2^{\Pi}$ for all $i = 0, 1, \ldots$ is a trace of a trajectory $\mathbf{x}$ if there exists an associated time sequence $t_0, t_1, \ldots$ of time instants such that*

1. $t_0 = 0$

2. $t_\tau \to \infty$ *as* $\tau \to \infty$

3. $t_i < t_{i+1}$

4. $x(t_i) \in [\![P_i]\!]$

5. *if* $P_i \neq P_{i+1}$, *then there exists some* $t_i' \in [t_i, t_{i+1}]$ *such that* $x(t) \in [\![P_i]\!]$ *for all* $t \in (t_i, t_i')$, $x(t) \in [\![P_{i+1}]\!]$ *for all* $t \in (t_i', t_{i+1})$, *and either* $x(t_i') \in [\![P_i]\!]$ *or* $x(t_i') \in [\![P_{i+1}]\!]$.

The trace of the system trajectory gives the sequence of atomic propositions satisfied by the system, and thus bridges the system behavior with temporal logic specification. Given a controller μ, we denote the trajectory under controller μ as $\mathbf{x}^\mu$. The trace of trajectory $\mathbf{x}^\mu$ is denoted by $Trace(\mathbf{x}^\mu)$. Suppose a specification φ belonging to $\text{LTL}_{\setminus\bigcirc}$ is given to system (9.1). If $Trace(\mathbf{x}^\mu) \models \varphi$, we say system (9.1) satisfies φ under controller μ, or controller μ satisfies φ. We state the problem of interest as follows.

Problem 9.1. *Compute a feedback controller* $\mu : \mathcal{X} \times [0, \infty) \mapsto \mathcal{U}$ *under which system* (9.1) *satisfies the given LTL specification* φ *belonging to* $LTL_{\setminus\bigcirc}$. *That is, compute a controller* μ *such that* $Trace\,(\mathbf{x}^\mu) \models \varphi$.

9.4 Abstraction-Free Control Synthesis using Control Barrier Functions

This section presents a framework to solve Problem 9.1. We first introduce two types of CBFs by extending Definition 9.1 and 9.2. Given an LTL specification φ, we then present how to design CBFs using the automaton of the LTL specification. We construct a sequence of LTL formulas that correspond to an accepting run on the DRA of the LTL specification. Then we define a time varying CBF for each formula. We show that satisfying each formula is equivalent to guaranteeing the positivity of the corresponding CBF. Then we compute the controllers that ensure the CBF associated with each formula to be positive, and hence satisfies the LTL specification.

9.4.1 Time Varying ZCBF and FCBF

In the following, we introduce time varying zeroing CBF (ZCBF) [134] and finite time convergence CBF (FCBF) by extending Definition 9.1 and 9.2.

Definition 9.4 (Time Varying Zeroing CBF (ZCBF) [134])**.** *Consider a dynamical system* (9.1) *and a continuously differentiable function* $h : \mathcal{X} \times [0, \infty) \mapsto \mathbb{R}$. *If there exists a locally Lipschitz extended class* $\mathcal{K}$ *function* α *such that for all* $x(t) \in \mathcal{X}$ *the following inequality holds*

$$\sup_{u \in \mathcal{U}} \left\{ \frac{\partial h(x,t)}{\partial x} f(x) + \frac{\partial h(x,t)}{\partial x} g(x) u + \frac{\partial h(x,t)}{\partial t} + \alpha(h(x,t)) \right\} \geq 0, \tag{9.6}$$

then function h *is a ZCBF.*

Given a ZCBF h, the set of controllers satisfying Eqn. (9.6) is represented as $\mathcal{U}_Z(x,t) = \{\mu | \frac{\partial h(x,t)}{\partial x} f(x) + \frac{\partial h(x,t)}{\partial x} g(x)\mu(x,t) + \frac{\partial h(x,t)}{\partial t} + \alpha(h(x,t)) \geq 0\}$. The following proposition [134] characterizes $\mathcal{U}_Z(x,t)$.

Proposition 9.1. *Let $\mathcal{C}(t) = \{x | h(x,t) \geq 0\}$, where $h : \mathcal{X} \times [0,\infty) \mapsto \mathbb{R}$. Consider a feedback controller $\mu(x,t) \in \mathcal{U}_Z(x,t)$. If h is a ZCBF, then for all $x \in \mathcal{C}(t)$ and $t \geq 0$, $\mu(x,t)$ guarantees the set $\mathcal{C}(t)$ to be forward invariant.*

Motivated by FCBF in Definition 9.2, we define time varying FCBF as follows.

Definition 9.5 (Time Varying Finite Time Convergence CBF (FCBF))**.** *Consider a dynamical system* (9.1) *and a continuously differentiable function $h : \mathcal{X} \times [0,\infty) \mapsto \mathbb{R}$. If there exist constants $\rho \in [0,1)$ and $\gamma > 0$ such that for all $x(t) \in \mathcal{X}$ the following inequality holds*

$$\sup_{u \in \mathcal{U}} \left\{ \frac{\partial h(x,t)}{\partial x} f(x) + \frac{\partial h(x,t)}{\partial x} g(x) u + \frac{\partial h(x,t)}{\partial t} + \gamma \cdot sgn(h(x,t)) |h(x,t)|^\rho \right\} \geq 0, \tag{9.7}$$

then function h is a FCBF.

Given an FCBF h, the set of controllers that satisfy (9.7) is represented as $\mathcal{U}_F(x,t) = \{\mu | \frac{\partial h(x,t)}{\partial x} f(x) + \frac{\partial h(x,t)}{\partial x} g(x)\mu(x,t) + \frac{\partial h(x,t)}{\partial t} + \gamma \cdot \text{sgn}(h(x,t)) |h(x,t)|^\rho \geq 0\}$. The following proposition extends the result in [132] on time invariant FCBF and characterizes $\mathcal{U}_F(x,t)$.

Proposition 9.2. *Let $\mathcal{C}(t) = \{x | h(x,t) \geq 0\}$, where $h : \mathcal{X} \times [0,\infty) \mapsto \mathbb{R}$. Consider a feedback controller $\mu(x,t) \in \mathcal{U}_F(x,t)$. If h is an FCBF, then for any initial state $x_0 \in \mathcal{X}$, controller $\mu(x,t)$ guarantees that the system will be steered to the set $\mathcal{C}(t)$ within finite time $0 < T < \infty$ such that $x(T) \in \mathcal{C}(T)$. The convergence time $T = \frac{|h(x_0,0)|^{(1-\rho)}}{\gamma(1-\rho)}$. Moreover, the system remains in $\mathcal{C}(t')$ for all $t' \geq T$.*

Proof. The proof follows [132]. Construct a Lyapunov function $V(x,t) = \max\{-h(x,t), 0\}$. We can verify that $V(x,t) = 0$ for all $x(t) \in \mathcal{C}(t)$, $V(x,t) > 0$ for all $x(t) \in \mathcal{X} \setminus \mathcal{C}(t)$, and $\frac{\mathrm{d}}{\mathrm{d}t} V(x,t) \leq \gamma V(x,t)^\rho$. By Theorem 4.1 in [135], finite time stability holds for system (9.1). Thus if $x_0 \in \mathcal{C}(t)$, controllers in $\mathcal{U}_F(x,t)$ render $\mathcal{C}(t)$ forward invariant. If $x_0 \notin \mathcal{C}(t)$, then the system converges to $\mathcal{C}(t)$ within finite time $T = \frac{|h(x_0,0)|^{(1-\rho)}}{\gamma(1-\rho)}$. □

9.4.2 Design of Control Barrier Functions

In this subsection, we first construct a sequence of LTL formulas so that satisfying all formulas is equivalent to satisfying the given LTL formula φ. Then we show how to design time varying CBFs for each formula.

Given an LTL specification φ, we compute the DRA associated with φ, and pick an accepting run $\eta = q_0 \dots q_J, (q_{J+1} \dots q_{J+N})^\omega$ of the DRA. Although a similar idea is used in [45, 46], in which a sequence of reachability problems is selected, as we will show later, our proposed CBF design enhances the feasibility of control synthesis. The complexity of

constructing the DRA is doubly exponential in the size of the formula in the worst-case as in the existing works on both abstraction-free and abstraction-based LTL synthesis [26, 39, 133]. However, we note that several important classes of LTL formulas have DRAs of polynomial size [136], and that our approach mitigates the exponential complexity of computing a finite state abstraction.

Rewriting η into prefix-suffix form, we have that the sequence of states $q_0 \dots q_J$ forms the prefix η_{pref}, and the sequence of states $q_{J+1} \dots q_{J+N}$ forms the suffix η_{suff}. We denote the transition from state q_j to q_{j+1} as η_j, and denote the input word of transition η_j as ϕ_j. The input word ϕ_j is in the form of conjunction or disjunction of atomic propositions [18], i.e., $\phi_j = \pi_1 \bowtie \dots \bowtie \pi_k$ where $\pi_i \in \Pi$ is an atomic proposition for all $i = 1, \dots, k$ and $\bowtie \in \{\wedge, \vee\}$.

Given the accepting run η, we construct a sequence of formulas $\{\psi_j | j = 0, 1, \dots, J+N\}$, where ψ_j corresponds to transition η_j as follows. We denote the input word corresponding to the self-transition at state q_j as Φ_j, i.e., $\delta(q_j, \Phi_j) = q_j$. We also note that ϕ_j is the input word corresponding to a transition from q_j to q_{j+1}. We then construct a formula ψ_j corresponding to transition η_j as

$$\psi_j = \Phi_j \ \mathrm{U} \ \Box \left(\phi_j \wedge \Phi_{j+1}\right). \tag{9.8}$$

Formula ψ_j indicates that no transition starting from state q should occur except self-transition and η_j. Since both prefix and suffix of η are over finite horizon, only a finite number of formulas $\{\psi_j | j = 0, 1, \dots, J+N\}$ are generated.

We then assign a sequence of time instants $t_1 < \dots < t_J$ as the deadlines of each transition $\eta_0, \eta_1, \dots, \eta_J$ of η_{pref}. The deadlines of the transitions of the suffix can be generated as $n\Delta + t_{J+1}$, where n is a nonnegative integer and $\Delta \geq 0$. We additionally let $t_0 = 0 < t_1$. We define the active time of each formula ψ_j as $[t_j, t_{j+1}]$, during which formula ψ_j must be satisfied. There are two advantages of defining the active time of each formula ψ_j. First, although each formula ψ_j needs to be interpreted over infinite runs, the active time enables us to interpret each ψ_j over finite runs. That is, formula ψ_j needs to be satisfied during $[t_j, t_{j+1}]$. For time $t > t_{j+1}$, formula ψ_j can be violated. Second, the active time allows our approach to satisfy multiple, sequential constraints (e.g., reaching disjoint regions A and B) that cannot be satisfied simultaneously.

Given a time interval $[t, t']$ and controller μ, we let $\mathbf{x}^\mu([t, t'])$ be the system trajectory during time interval $[t, t']$ under controller μ. The trace of $\mathbf{x}^\mu([t, t'])$ is denoted as $Trace\left(\mathbf{x}^\mu([t, t'])\right)$. Denote the system state under controller μ at time t as $\mathbf{x}^\mu(t)$. We then show the effectiveness of sub-formula (9.8) by analyzing the relationship between satisfying each formula ψ_j and run η.

Lemma 9.4. *Let η be an accepting run and ψ_j be a formula in the form of Eqn. (9.8) whose active time is $[t_j, t_{j+1}]$. If $Trace\left(\mathbf{x}^\mu([0, t_j])\right)$ steers the DRA from q_0 to q_j, and $Trace\left(\mathbf{x}^\mu([t_j, t_{j+1}])\right) \models \psi_j$, then the DRA transitions from state q_j to q_{j+1} during time interval $[t_j, t_{j+1}]$. Moreover, the DRA remains in state q_{j+1} until at least time t_{j+1}.*

Proof. We first prove that if $Trace\,(\mathbf{x}^\mu([0,t_j]))$ steers the DRA from q_0 to q_j, and during $[t_j,t_{j+1}]$ the relation $Trace\,(\mathbf{x}^\mu([t_j,t_{j+1}])) \models \psi_j$ holds, then the DRA transitions from state q_j to q_{j+1} during time interval $[t_j,t_{j+1}]$. We prove by contradiction. Suppose the current state of the DRA is q_j and $Trace\,(\mathbf{x}^\mu([t_j,t_{j+1}])) \models \psi_j$, while the DRA transitions from state q_j to some state $q' \neq q_{j+1}$. By the semantics of until operator U, there must exist some time $t \in [t_j,t_{j+1}]$ such that $L(\mathbf{x}^\mu(t')) \models \phi_j \wedge \Phi_{j+1}$ for all $t' \in [t,t_{j+1}]$ in order to make $Trace\,(\mathbf{x}^\mu([t_j,t_{j+1}])) \models \psi_j$ hold. Then by the semantics of and operator $\wedge$, $L(\mathbf{x}^\mu(t)) \models \phi_j \wedge \Phi_{j+1}$ implies that $L(\mathbf{x}^\mu(t)) \models \phi_j$. Since ϕ_j is the input associated with the transition from q_j to q_{j+1}, then $q' = q_{j+1}$. Otherwise, the DRA contains nondeterminism which conflicts Definition 3.5.

We then prove that the DRA remains in q_{j+1} until at least time t_{j+1}. Suppose the DRA transitions from q_{j+1} to some state q before t_{j+1}. This is equivalent to the fact that there exist some state $q \in \mathcal{N}(q_{j+1})$ and time $t \in [t_j,t_{j+1}]$ such that $L(\mathbf{x}^\mu(t)) \models \phi^q_{j+1}$, where ϕ^q_{j+1} is the input word associated with transition from state q_{j+1} to some neighbor state q. However, this contradicts Φ_{j+1}, and thus the DRA cannot transition to some state $q \in \mathcal{N}(q_{j+1})$. By the definition of neighbor states $\mathcal{N}(q_{j+1})$, the DRA can only take the self-transition at q_{j+1}. □

Inducting the results on Lemma 9.4 yields Corollary 9.1.

Corollary 9.1. *If* $Trace\,(\mathbf{x}^\mu([t_j,t_{j+1}])) \models \psi_j$ *for all* $j = 0,1,\ldots,$ *then* $Trace\,(\mathbf{x}^\mu) \models \varphi$.

Lemma 9.4 and Corollary 9.1 imply that, in order to ensure that the specification is satisfied, it suffices to ensure that the trajectory under controller μ satisfies each ψ_j within its active time $[t_j,t_{j+1}]$. In what follows, we construct a set of CBFs that will be used to ensure satisfaction of each ψ_j.

In the following, we design CBFs for Φ_j and $\phi_j \wedge \Phi_{j+1}$. We first define a CBF h_π for each atomic proposition π that is involved in ψ_j. We consider CBFs in the form of $h_\pi(x,t) = M_\pi(t) + Z_\pi(x)$ for all π, where $M_\pi(t)$ and $Z_\pi(x)$ are called guard function and state function, respectively. The state function $Z_\pi(x)$ is a function of state x that captures if the state x is in $[\![\pi]\!]$, i.e., $[\![\pi]\!] = \{x | Z_\pi(x) \geq 0\}$. The guard function $M_\pi(t) = \frac{E_\pi}{1+e^{-b_\pi(t+c_\pi)}} - \epsilon_\pi$ is a logistic function, where $E_\pi > 0$, $b_\pi > 0$, and $\epsilon_\pi \geq 0$. The guard function $M_\pi(t)$ is introduced so that each atomic proposition π, and hence ψ_j, only need to be satisfied during their active time.

We then show how to choose E_π, b_π, c_π, and ϵ_π for each π. First, if atomic proposition π is satisfied at time $t = 0$, then $h_\pi(x_0,0) \geq 0$. If π is not satisfied at time $t = 0$, then $h_\pi(x_0,0) < 0$. These two requirements are captured by Eqn. (9.9a) and (9.9b). Second, given the deadline t_j of atomic proposition π, we have $M_\pi(t_j) \leq 0$, as shown in Eqn. (9.9c).

To summarize, we have the following inequalities:

$$\frac{E_\pi}{1+e^{-b_\pi c_\pi}} - \epsilon_\pi + Z_\pi(x_0) \geq 0, \text{ if } \pi \in L(x_0) \tag{9.9a}$$

$$\frac{E_\pi}{1+e^{-b_\pi c_\pi}} - \epsilon_\pi + Z_\pi(x_0) < 0, \text{ if } \pi \notin L(x_0) \tag{9.9b}$$

$$\frac{E_\pi}{1+e^{-b_\pi (t_j + c_\pi)}} - \epsilon_\pi \leq 0, \tag{9.9c}$$

$$E_\pi > 0, b_\pi > 0, \epsilon_\pi \geq 0 \tag{9.9d}$$

Inequalities (9.9) are solved as follows. We first pick some $b_\pi > 0$ and c_π such that $c_\pi \leq t_{j+1}$ if π is involved in Φ_j or ϕ_j, and $c_\pi \leq t_{j+2}$ if π is involved in Φ_{j+1}. Fixing the values of b_π and c_π, then E_π and ϵ_π can be obtained by solving the linear inequalities (9.9). We characterize the CBFs obtained by solving (9.9) using the following lemma.

Lemma 9.5. *Let t_j be the deadline of atomic proposition π, and h_π be the CBF obtained by solving* (9.9)*. For any $t \leq t_j$, if $h_\pi(x,t) \geq 0$, then $x(t) \in [\![\pi]\!]$.*

Proof. Inequality (9.9c) indicates that $M_\pi(t_j) \leq 0$, where t_j is the deadline of π. By Eqn. (9.9d), the guard function $M_\pi(t)$ is monotone increasing. Therefore, for all $t \leq t_j$, $M_\pi(t) \leq 0$ holds. By the definition of $h_\pi(x,t)$, we have that $Z_\pi(x) = h_\pi(x,t) - M_\pi(t)$. Given $M_\pi(t) \leq 0$ for all $t \leq t_j$ and $h_\pi(x,t) \geq 0$, we have $Z_\pi(x) \geq 0$, and thus the system state $x(t)$ is in the region $\{x(t) | Z_\pi(x(t)) \geq 0\} = [\![\pi]\!]$. □

Given a CBF h_π for each atomic proposition π that is involved in ψ_j, we compute the CBFs for Φ_j and $\phi_j \wedge \Phi_{j+1}$. We note that Φ_j and $\phi_j \wedge \Phi_{j+1}$ are both in the forms of conjunctions/disjunctions of atomic propositions [18]. We utilize the following definition to construct the CBF for Φ_j and $\phi_j \wedge \Phi_{j+1}$.

Definition 9.6. *Consider a set of atomic proposition $\{\pi_i | i = 1, \ldots, k\}$. Let $h_{\pi_i} : \mathbb{R}^n \times [0,\infty) \mapsto \mathbb{R}$ be the CBF of each π_i defined as $h_{\pi_i}(x,t) = M_{\pi_i}(t) + Z_{\pi_i}(x)$ for each atomic proposition π_i, where $M_{\pi_i}(t)$ is computed by* (9.9) *and $Z_{\pi_i}(t)$ is defined as $\{x | Z_{\pi_i}(x) \geq 0\} = [\![\pi_i]\!]$. Consider a formula $\phi' = \pi_1 \bowtie \ldots \bowtie \pi_{k-1}$, where $\bowtie \in \{\wedge, \vee\}$. Let $h_{\phi'}$ be the CBF of ϕ'. Then the CBF of formula $\phi = \phi' \wedge \pi_k$ is*

$$h_\phi(x,t) = -\ln\left[\exp\left(-h_{\phi'}(x,t)\right) + \exp\left(-h_{\pi_k}(x,t)\right)\right]. \tag{9.10}$$

The CBF h_ϕ of formula $\phi = \phi' \vee \pi_k$ for some $\lambda > 0$ is

$$h_\phi(x,t) = \frac{h_{\phi'}(x,t) e^{\lambda h_{\phi'}(x,t)} + h_{\pi_k}(x,t) e^{\lambda h_{\pi_k}(x,t)}}{e^{\lambda h_{\phi'}(x,t)} + e^{\lambda h_{\pi_k}(x,t)}}. \tag{9.11}$$

Definition 9.6 recursively defines the CBF for a formula ϕ in the form of $\phi = \pi_1 \bowtie \ldots \bowtie \pi_k$, where $\bowtie \in \{\wedge, \vee\}$. By Lemma 9.1, we have that CBFs (9.10) and (9.11) bound

$\min\{h_{\phi'}(x,t), h_{\pi_k}(x,t)\}$ and $\max\{h_{\phi'}(x,t), h_{\pi_k}(x,t)\}$ from below, respectively. When Φ_j and $\phi_j \wedge \Phi_{j+1}$ are in the forms of $\pi_1 \bowtie \ldots \bowtie \pi_k$, where $\bowtie \in \{\wedge, \vee\}$, their CBFs can be obtained by recursively applying Definition 9.6.

Algorithm 18 Algorithm for computing the CBFs for each formula ψ_j.

1: **Input**: LTL specification φ
2: **Output:** CBFs for each formula ψ_j
3: Compute the DRA associated with LTL specification φ, and the set of accepting runs on the DRA.
4: Pick an accepting run η on the DRA, and identify each formula ψ_j associated with each transition η_j in η as (9.8).
5: Specify a sequence of time $0 < t_1 < \ldots$ for accepting run η.
6: Pick a set of feasible coefficients for relations (9.9) for each atomic proposition π involved in ψ_j.
7: Recursively compute CBFs for Φ_j and $\phi_j \wedge \Phi_{j+1}$ using Definition 9.6.
8: **return** CBFs for Φ_j and $\phi_j \wedge \Phi_{j+1}$

The procedure we used to design the CBFs for each ψ_j is presented in Algorithm 18. We characterize the construction of CBFs for Φ_j and $\phi_j \wedge \Phi_{j+1}$ using the following proposition.

Lemma 9.6. *Let h_ϕ be the CBF obtained by Algorithm 18, where $\phi \in \{\Phi_j, \phi_j \wedge \Phi_{j+1}\}$. For any time $t \in [t_j, t_{j+1}]$, if $h_\phi(x,t) \geq 0$, then $L(x(t)) \models \phi$.*

Proof. We prove by induction. Consider $\phi = \pi_1 \wedge \pi_2$. Then h_ϕ is computed as (9.10). By Lemma 9.1, $h_\phi(x,t) \geq 0$ implies that $h_{\pi_1}(x,t) \geq 0$ and $h_{\pi_2}(x,t) \geq 0$. By Lemma 9.5, $x(t) \in [\![\pi_1]\!] \cap [\![\pi_2]\!]$, and thus ϕ is satisfied. Consider $\phi = \pi_1 \vee \pi_2$. Then h_ϕ is computed as (9.11). By Lemma 9.1, $h_\phi(x,t) \geq 0$ implies that $h_{\pi_1}(x,t) \geq 0$ or $h_{\pi_2}(x,t) \geq 0$. By Lemma 9.5, $x(t) \in [\![\pi_1]\!] \cup [\![\pi_2]\!]$, and thus ϕ is satisfied. These two cases serve as our induction base.

Suppose the lemma holds after applying Eqn. (9.10) and (9.11) $k-1$ times, and denote the corresponding CBF as h_ϕ^{k-1}. If the k-th operation is a conjunction with atomic proposition π, then $h_\phi^k(x,t)$ is obtained by (9.10). By Lemma 9.1, $h_\phi^k(x,t) \geq 0$ implies that $h_\pi(x,t) \geq 0$ and $h_\phi^{k-1}(x,t) \geq 0$. By Lemma 9.5, $h_\pi(x,t) \geq 0$ implies $x(t) \in [\![\pi]\!]$. By our inductive hypothesis, $h_\phi^{k-1}(x,t) \geq 0$ implies $L(x(t)) \models \phi$ after applying Eqn. (9.10) and (9.11) $k-1$ times. Combining the arguments above, we have $L(x(t)) \models \phi$ when the k-th operator is a conjunction. If the k-th operation is a disjunction with atomic proposition π, then $h_\phi^k(x,t)$ is obtained by Eqn. (9.11). By Lemma 9.1, $h_\phi^k(x,t) \geq 0$ implies that $h_\pi(x,t) \geq 0$ or $h_\phi^{k-1}(x,t) \geq 0$. Similar to the conjunction case, we can conclude $L(x(t)) \models \phi$ when the k-th operator is a disjunction. Since ϕ is in the form of conjunction and disjunction of finite number of atomic propositions, the lemma holds by induction. □

9.4.3 CBF-based Controller Synthesis

In this subsection, we present how to compute the controllers that satisfy each formula ψ_j in the form of Eqn. (9.8). We then show that by satisfying all formulas ψ_j for all j, the LTL specification φ is satisfied.

Lemma 9.7. *Consider a formula ψ_j in the form of (9.8) whose active time is $[t_j, t_{j+1}]$. Let h_{Φ_j} and h_{Ω_j} be the CBFs of Φ_j and $\phi_j \wedge \Phi_{j+1}$ obtained using Algorithm 18, respectively. Then for all $t \in [t_j, t_{j+1}]$ any feedback controller in*

$$\mathcal{U}_{\psi_j}(x,t) = \begin{cases} \mathcal{U}_2(x,t), & \text{if } L(x(t)) \models \phi_j \wedge \Phi_{j+1}, \\ \mathcal{U}_1(x,t) \cap \mathcal{U}_2(x,t), & \text{if } L(x(t)) \models \Phi_j, \\ \emptyset, & \text{otherwise} \end{cases}$$

satisfies $Trace\,(\mathbf{x}^{\mu}([t_j, t_{j+1}])) \models \psi_j$, where

$$\begin{aligned} \mathcal{U}_1(x,t) = \Big\{ \mu \Big| & \frac{\partial h_{\Phi_j\Omega_j}(x,t)}{\partial x} f(x) + \frac{\partial h_{\Phi_j\Omega_j}(x,t)}{\partial x} g(x)\mu(x,t) \\ & + \frac{\partial h_{\Phi_j\Omega_j}(x,t)}{\partial t} + \alpha(h_{\Phi_j\Omega_j}(x,t)) \geq 0 \Big\}, \\ \mathcal{U}_2(x,t) = \Big\{ \mu \Big| & \frac{\partial h_{\Omega_j}(x,t)}{\partial x} f(x) + \frac{\partial h_{\Omega_j}(x,t)}{\partial x} g(x)\mu(x,t) \\ & + \frac{\partial h_{\Omega_j}(x,t)}{\partial t} + \gamma \cdot sgn(h_{\Omega_j}(x,t)) |h_{\Omega_j}(x,t)|^{\rho} \geq 0 \Big\} \\ h_{\Phi_j\Omega_j}(x,t) = & \frac{h_{\Phi_j}(x,t) e^{\lambda h_{\Phi_j}(x,t)} + h_{\Omega_j}(x,t) e^{\lambda h_{\Omega_j}(x,t)}}{e^{\lambda h_{\Phi_j}(x,t)} + e^{\lambda h_{\Omega_j}(x,t)}}. \end{aligned}$$

Proof. By the semantics of until operator U and Definition 9.3, it suffices to show that there exists a time sequence t_j, T, t_{j+1} such that (i) $t_j < t_{j+1}$, (ii) $T \in [t_j, t_{j+1}]$, and (iii) $L(x(t)) \models \Phi_j$ for all $t \in [t_j, T)$ and $L(x(t)) \models \phi_j \wedge \Phi_{j+1}$ for all $t \in [T, t_{j+1}]$. We show that each of these conditions is satisfied.

First, condition $t_j < t_{j+1}$ holds by the construction of deadlines. To guarantee $T \in [t_j, t_{j+1}]$, we can tune parameters ρ and γ as given in Proposition 9.2 so that $T \leq t_{j+1}$. In the following, we show that there exists some time $T \in [t_j, t_{j+1}]$ such that $L(x(t)) \models \Phi_j$ for all $t \in [t_j, T)$ and $L(x(t)) \models \phi_j \wedge \Phi_{j+1}$ for all $t \in [T, t_{j+1}]$.

We start with the case where $L(x(t)) \models \phi_j \wedge \Phi_j$. In this case, $T = t$ and we need to guarantee $L(x(t')) \models \phi_j \wedge \Phi_{j+1}$ for all $t' \in [t, t_{j+1}]$ so that formula ψ_j is satisfied. By Proposition 9.2, the set of controllers $\mathcal{U}_2(x,t)$ ensures that the system remains in the set $\{x | h_{\Omega_j}(x,t') \geq 0\}$ for all $t' \geq t$. By Lemma 9.6, we have that $L(x(t')) \models \phi_j \wedge \Phi_j$ for all $x(t') \in \{x(t) | h_{\Omega_j}(x,t') \geq 0\}$ for all $t' \in [t, t_{j+1}]$. Therefore, $Trace\,(\mathbf{x}^{\mu}([t, t_{j+1}])) \models \psi_j$ for all $t \in [t_j, t_{j+1}]$ in this case.

We then consider the case where $L(x(t)) \models \Phi_j$. By Proposition 9.1, the set of controllers

$\mathcal{U}_1(x,t)$ guarantees that the system remains in the set $\{x(t')|h_{\Phi_j\Omega_j}(x,t') \geq 0\}$ for all $t' \in [t, t_{t+1}]$. By Lemma 9.6, we have that $L(x(t')) \models \Phi_j$, or $L(x(t')) \models \phi_j \wedge \Phi_{j+1}$, or both, for all $x(t') \in \{x(t')|h_{\Phi_j\Omega_j}(x,t') \geq 0\}$ where $t' \in [t, t_{j+1}]$. By Proposition 9.2, the set of controllers $\mathcal{U}_2(x,t)$ ensures that the system will be steered to the set $\{x|h_{\Omega_j}(x,t) \geq 0\}$ at some time $T \geq t$ if $x(t) \notin \{x|h_{\Omega_j}(x,t) \geq 0\}$. Moreover, the system remains in $x(t') \in \{x|h_{\Omega_j}(x,t') \geq 0\}$ for all $t' \in [T, t_{j+1}]$. By Lemma 9.6, we have that $Trace\,(\mathbf{x}^\mu([t, t_{j+1}])) \models \square(\phi_j \wedge \Phi_{j+1})$ for all $t \in [t_j, t_{j+1}]$. Thus, the controllers in $\mathcal{U}_1(x,t) \cap \mathcal{U}_2(x,t)$ guarantee $Trace\,(\mathbf{x}^\mu([t, t_{j+1}])) \models \psi_j$ for all $t \in [t_j, t_{j+1}]$.

Combining the arguments above, we have that the controllers in $\mathcal{U}_{\psi_j}(x,t)$ satisfy that $Trace\,(\mathbf{x}^\mu([t_j, t_{j+1}])) \models \psi_j$. □

We note that the computation of controllers can be simplified when the formula ψ_j is in some simple forms. Consider a formula ψ_j as per (9.8). When $\Phi_j = True$, then $\psi_j = \square(\phi_j \wedge \Phi_{j+1})$. The controllers satisfying ψ_j in this case are given by the following corollary.

Corollary 9.2. *Suppose $\psi_j = \square\phi$. Let h_ϕ be the CBF of ϕ that is obtained using Algorithm 18. Then any feedback controller in*

$$\mathcal{U}_{\psi_j}(x,t) = \begin{cases} \left\{\mu | \frac{\partial h_\phi(x,t)}{\partial x} f(x) + \frac{\partial h_\phi(x,t)}{\partial x} g(x)\mu(x,t) + \frac{\partial h_\phi(x,t)}{\partial t} + \alpha(h_\phi(x,t)) \geq 0\right\}, \text{ if } L(x_0) \models \phi \\ \emptyset, \text{ otherwise} \end{cases}$$

satisfies ψ_j.

Proof. The proof follows by replacing Φ_j in Lemma 9.7 with unconditionally true. □

We finally show that by satisfying each formula ψ_j, specification φ is satisfied.

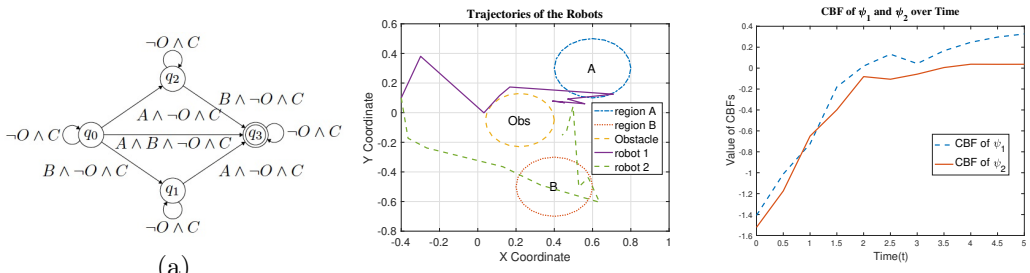
Theorem 9.1. *Applying the controllers in $\mathcal{U}_{\psi_j}(x,t)$ for all $t \geq 0$ as given in Lemma 9.7 renders $Trace\,(\mathbf{x}^\mu) \models \varphi$.*

Proof. Applying μ for each time interval $[t_j, t_{j+1}]$, we have that $Trace\,(\mathbf{x}^\mu([t_j, t_{j+1}])) \models \psi_j$ due to Lemma 9.7. Then according to Lemma 9.4 and Corollary 9.1, we have that satisfying the sequence of formulas ψ_j for all j is equivalent to executing the run η on the DRA. Since η is an accepting run, we can conclude that $Trace\,(\mathbf{x}^\mu) \models \varphi$. □

Using the treatment in [137, 44], we can compute the control input $u(t)$ at each time t by solving the quadratic program (QP):

$$\begin{aligned} \min_u \quad & u^\top R u \\ \text{s.t.} \quad & u(t) \in \mathcal{U}_{\psi_j}(x,t),\ \forall \psi_j \end{aligned}$$

where $R \in \mathbb{R}^{m \times m}$ is a positive semi-definite matrix that quantifies the cost of control, and $\mathcal{U}_{\psi_j}(x,t)$ is given by Lemma 9.7. The constraint set of this QP requires that the CBF



(a) (b) (c)

Figure 9-1: Fig. 9-1a shows the DRA associated with $\varphi = \diamondsuit A \wedge \diamondsuit B \wedge \square(\neg O \wedge C)$. It has one Rabin pair $(\emptyset, \{q_3\})$. Fig. 9-1b presents the trajectories of both robot using the proposed approach in this chapter. The trajectory of the first robot is plotted in solid line, while the trajectory of the second robot is plotted in dotted line. The first robot eventually reaches region A, and the second robot eventually reaches region B. Both robots successfully avoid the obstacle region O and remain close enough to each other. CBFs h_{ψ_1} and h_{ψ_2} are presented in Fig. 9-1c. h_{ψ_1} is plotted in dotted line and h_{ψ_2} is plotted in solid line. Both CBFs become positive at some time $t \geq 0$, at which the formulas corresponding to the CBFs are satisfied.

constraints as given in Lemma 9.7 must be satisfied for each formula ψ_j. We note that the deadlines of formulas ψ_j's corresponding to transitions in η_{suff} are generated periodically as $n\Delta + t_{J+1}$. We can impose the constraint $u(t) \in \mathcal{U}_{\psi_i}(x,t)$ for $i \in \{j, \ldots, j+J+N\}$, resulting in a finite number of constraints in the QP. The controllers for future time can then be generated by implementing the controllers periodically.

9.5 Case Study

In this section, we present a numerical case study on a two robot homogeneous multi-agent system. Consider a multi-agent system consisting of two robots whose dynamics are given as $\dot{x} = u$, where $x \in \mathbb{R}^4$ is the system state, and $u \in \mathbb{R}^4$ is the input. The state variables x_1 and x_2 give the coordinate of the first robot, and state variables x_3 and x_4 give the coordinate of the second robot. The initial positions for both robots are $[-0.4, 0.1]$.

Each of the robots has its respective goal region, denoted as region A and region B. Both robots are required to eventually reach their goal regions. In the meantime, both robots need to avoid the obstacle region, denoted as O. Furthermore, the robots keep exchanging information with each other, and hence must remain close enough. The tasks can be represented by an LTL formula $\varphi = \diamondsuit A \wedge \diamondsuit B \wedge \square(\neg O \wedge C)$, where C represents the connectivity specification. In this case study, region A is modeled as $\{x | Z_A(x) \geq 0\}$ where $Z_A(x) = 0.2 - \|[x_1, x_2] - [0.6, 0.3]\|_2$. Region B is modeled as $\{x | Z_B(x) \geq 0\}$ where $Z_B(x) = 0.2 - \|[x_3, x_4] - [0.4, -0.5]\|_2$. The obstacle region is modeled as $\{x | Z_O(x) \geq 0\}$

where $Z_O(x) = 0.18 - \|x - [0.22, -0.05, 0.22, -0.05]\|_2$. The connectivity between the two robots is given as $C = \{x | Z_C(x) \geq 0\}$ where $Z_C(x) = \sqrt{x_3 + 0.39} - \|[x_1, x_2] - [x_3, x_4]\|_2$.

We compare the proposed approach with the approach proposed in [46]. Since there is single CBF for each region of interest, the relaxation proposed in [46] coincides with the traditional CBF-based approach. We observe that no feasible trajectory is synthesized since there is no feasible solution to the QP. The infeasibility of the QP is caused by the fact that the connectivity constraint requires the robots to stay close to each other, while steering the robots into region A and B makes them violate the connectivity constraint. This infeasibility agrees with the example presented in [45].

In the following, we demonstrate our proposed approach. Given the LTL specification φ, the DRA representing φ is shown in Fig. 9-1a. The accepting runs of the DRA include $q_0 q_1 (q_3)^\omega$, $q_0 (q_3)^\omega$, and $q_0 q_2 (q_3)^\omega$. We pick the run $\eta = q_0 q_1 (q_3)^\omega$ in this case study. Transition from q_0 to q_1 of the accepting run η corresponds to formula $\psi_0 = (\neg O \wedge C) \mathrm{U} \Box (B \wedge \neg O \wedge C)$. Transition from q_1 to q_2 corresponds to formula $\psi_1 = (\neg O \wedge C) \mathrm{U} \Box (A \wedge \neg O \wedge C)$. Self transition at q_3 corresponds to formula $\psi_3 = \Box(\neg O \wedge C)$. Next, we assign the active time for each formulas during which the formula needs to be satisfied. In this case study, we let ψ_1 be satisfied during $[0, 2]$, and let ψ_2 be satisfied during $[2, 4]$. Using Eqn. (9.9), we then construct the CBFs for atomic propositions B and A as $h_B(x,t) = \frac{1}{1+e^{-(t-1.5)}} - 0.63 + 0.2 - \|[x_1, x_2] - [0.4, -0.5]\|_2$, $h_A(x,t) = \frac{1}{1+e^{-(t-0.5)}} - 0.9 + 0.2 - \|[x_1, x_2] - [0.6, 0.3]\|_2$, respectively. Then the CBF for each formula can be constructed by Definition 9.6. For instance, $h_{\psi_3}(x,t) = -\ln(\exp(-Z_O(x)) + \exp(-Z_C(x)))$.

In the following, we formulate the QP to solve for the controllers so that the LTL specification φ is satisfied. According to Lemma 9.7, we have the following constraints for each formula ψ_j:

$$\frac{\partial h_{\Phi_j \Omega_j}(x,t)}{\partial x} f(x) + \frac{\partial h_{\Phi_j \Omega_j}(x,t)}{\partial x} g(x) \mu(x,t) + \frac{\partial h_{\Phi_j \Omega_j}(x,t)}{\partial t} + \alpha(h_{\Phi_j \Omega_j}(x,t)) \geq 0 \tag{9.12}$$

$$\frac{\partial h_{\Omega_j}(x,t)}{\partial x} f(x) + \frac{\partial h_{\Omega_j}(x,t)}{\partial x} g(x) \mu(x,t) + \frac{\partial h_{\Omega_j}(x,t)}{\partial t} + \gamma \cdot \mathrm{sgn}(h_{\Omega_j}(x,t)) |h_{\Omega_j}(x,t)|^\rho \geq 0 \tag{9.13}$$

where Φ_j and Ω_j are defined as given in Lemma 9.7, and

$$h_{\Phi_j \Omega_j}(x,t) = \frac{h_{\Phi_j}(x,t) e^{\lambda h_{\Phi_j}(x,t)} + h_{\Omega_j}(x,t) e^{\lambda h_{\Omega_j}(x,t)}}{e^{\lambda h_{\Phi_j}(x,t)} + e^{\lambda h_{\Omega_j}(x,t)}}.$$

Formulating (9.12) and (9.13) for all ψ_j's form the constraint set of the QP.

By solving the QP, we obtain the controllers for both robots. The trajectories of the robots are presented in Fig. 9-1b. We make the following observations. First, both robots eventually reach their target regions, while avoiding the obstacle region. Furthermore, the second robot reaches region B before the first robot reaches region A. This can be observed

from Fig. 9-1c. CBF h_{ψ_1} turns positive before h_{ψ_2}. Second, after the robots reaching their goal regions, they can still leave the goal region rather than remaining in the goal region. This is because our design of CBF adopts the guarding function $M_\psi(t)$ which increases over time and hence enhances the feasibility of the QP. The relaxation introduced by the guarding function enables the satisfaction of the connectivity constraint. As we could observe in Fig. 9-1c, CBFs h_{ψ_1} and h_{ψ_2} remain positive after the robots reaching their goal regions.

9.6 Conclusion

In this chapter, we studied the problem of control synthesis for CPS under LTL constraints modeled by LTL without next operator. We focused on synthesizing the controller to satisfy the LTL constraint without explicitly computing the abstraction of the CPS. A CBF-based approach is used in this chapter. We first constructed a sequence of LTL formulae corresponding to an accepting run on the DRA, and presented a design rule to design time-varying CBFs for the sequence of formulae. We introduced a function named guard function when designing CBFs, which enhances feasibility of CBF constraints. We showed that the positivity of the CBF implies the satisfaction of the LTL formula. Then we showed how to satisfy the set of formulae by guaranteeing the positivities of their CBFs. We formulated a QP to compute a controller that satisfies the LTL specification. A numerical case study is presented to illustrate the proposed approach.

We note that the approach proposed in this chapter is only applicable to CPS in the absence of the adversary. When there exists a malicious adversary, we need to generalize the definitions of ZCBF and FCBF to incorporate the impact from the adversary, which is subject to our future work.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 10

Conclusions and Future Work

In this thesis, we studied automatic and secure correct-by-construction control synthesis for cyber-physical systems operated in adversarial environments. We modeled the complex tasks assigned to the systems using temporal logic specifications. The adversary tampers with the system dynamics to manipulate the system state evolution and/or observation. We presented a sequence of problem settings along with their solution approaches.

We first presented the problem of maximizing the satisfaction probability when the system is given a linear temporal logic specification, as detailed in Chapter 4. We formulated the interaction between the controller and adversary as a stochastic game, and proposed a value iteration algorithm with convergence guarantee to compute the control policy.

Chapter 5 studied the problem of control synthesis to minimize the invariant constraint violation rate. We solved the problem by deriving the optimality condition leveraging optimal cost per stage problem, and developed a policy iteration algorithm to compute the controller's policy.

We studied the minimum violation control synthesis under a set of co-safe linear temporal logic specifications in Chapter 6. We also considered the limited observation capability of the adversary. We formulated the problem as a mixed integer nonlinear program. Two algorithms were proposed to solve the nonlinear program. The first algorithm converges to the optimal controller's policy with probability one, without any guarantee on the convergence rate. To this end, we proposed the second algorithm that approximately computes a feasible controller's policy.

In Chapter 7, we studied the control synthesis to maximize the satisfaction probability of a linear temporal logic specification under partially observable environments. The policies are represented using finite state controllers. We presented a value iteration algorithm to calculate the finite state controller with fixed size for the system. We also developed an algorithm to adaptively adjust the size of the finite state controller to improve the satisfaction probability.

Chapter 8 studied the problem of control synthesis under time-critical specifications modeled by metric interval temporal logic formulas. The adversary considered in this chapter

can not only tamper with the control input to manipulate the system dynamics, but also manipulate the time perception of the system to modify the timed system behavior. We presented a value iteration algorithm to compute a finite state controller for the system so that the satisfaction probability can be maximized.

Chapter 9 studied abstraction-free control synthesis for a continuous time control affine system. The linear temporal logic specification was decomposed into a sequence of sub-formulas and we used a set of control barrier function based constraints to compute the control input that guarantees the satisfaction of each sub-formula.

We note that the algorithms proposed in this dissertation are sound but not complete. Our future work is to develop algorithms with completeness guarantees. Additionally, we need to develop control barrier functions that are applicable to systems operated in adversarial environments, and apply them for abstraction-free control synthesis for systems under attacks. Finally, we will leverage the potential strength from learning algorithms to guarantee the satisfaction of temporal logic specifications for systems with unknown dynamics.

Bibliography

[1] R. Sharan, *Formal methods for control synthesis in partially observed environments: Application to autonomous robotic manipulation.* PhD thesis, California Institute of Technology, 2014.

[2] R. Baheti and H. Gill, "Cyber-physical systems," *The impact of control technology*, vol. 12, no. 1, pp. 161–166, 2011.

[3] S. Sridhar, A. Hahn, and M. Govindarasu, "Cyber–physical system security for the electric power grid," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 210–224, 2011.

[4] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost efficient resource management in fog computing supported medical cyber-physical system," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 1, pp. 108–119, 2015.

[5] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: the next computing revolution," in *Design Automation Conference*, pp. 731–736, IEEE, 2010.

[6] J. E. Sullivan and D. Kamensky, "How cyber-attacks in Ukraine show the vulnerability of the US power grid," *The Electricity Journal*, vol. 30, no. 3, pp. 30–35, 2017.

[7] A. Greenberg, "Hackers remotely kill a Jeep on the highway–with me in it," 2015.

[8] K. Koscher, S. Savage, F. Roesner, S. Patel, T. Kohno, A. Czeskis, D. McCoy, B. Kantor, D. Anderson, H. Shacham, *et al.*, "Experimental security analysis of a modern automobile," in *2010 IEEE Symposium on Security and Privacy*, pp. 447–462, IEEE Computer Society, 2010.

[9] Y. Shoukry, P. Martin, P. Tabuada, and M. Srivastava, "Non-invasive spoofing attacks for anti-lock braking systems," in *Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2013.

[10] H. Alemzadeh, D. Chen, X. Li, T. Kesavadas, Z. T. Kalbarczyk, and R. K. Iyer, "Targeted attacks on teleoperated surgical robots: Dynamic model-based detection and mitigation," in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 395–406, IEEE, 2016.

[11] J. P. Farwell and R. Rohozinski, "Stuxnet and the future of cyber war," *Survival*, vol. 53, no. 1, pp. 23–40, 2011.

[12] Y. Mo and B. Sinopoli, "False data injection attacks in control systems," in *Preprints of the 1st workshop on Secure Control Systems*, pp. 1–6, 2010.

[13] Y. Guan and X. Ge, "Distributed attack detection and secure estimation of networked cyber-physical systems against false data injection attacks and jamming attacks," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 1, pp. 48–59, 2017.

[14] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, "Unmanned aircraft capture and control via GPS spoofing," *Journal of Field Robotics*, vol. 31, no. 4, pp. 617–636, 2014.

[15] S. Karnouskos, "Stuxnet worm impact on industrial cyber-physical system security," in *IECON 2011-37th Annual Conference of the IEEE Industrial Electronics Society*, pp. 4490–4494, IEEE, 2011.

[16] R. Zhang and P. Venkitasubramaniam, "False data injection and detection in lqg systems: A game theoretic approach," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 1, pp. 338–348, 2019.

[17] S. Amin, A. A. Cárdenas, and S. S. Sastry, "Safe and secure networked control systems under denial-of-service attacks," in *International Workshop on Hybrid Systems: Computation and Control*, pp. 31–45, Springer, 2009.

[18] C. Baier, J.-P. Katoen, and K. G. Larsen, *Principles of Model Checking*. MIT Press, 2008.

[19] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[20] M. Kattenbelt and M. Huth, "Verification and refutation of probabilistic specifications via games," in *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2009.

[21] H. Fawzi, P. Tabuada, and S. Diggavi, "Secure estimation and control for cyber-physical systems under adversarial attacks," *IEEE Transactions on Automatic Control*, vol. 59, no. 6, pp. 1454–1467, 2014.

[22] M. H. Manshaei, Q. Zhu, T. Alpcan, T. Bacşar, and J.-P. Hubaux, "Game theory meets network security and privacy," *ACM Computing Surveys*, vol. 45, no. 3, p. 25, 2013.

[23] Q. Zhu and T. Basar, "Game-theoretic methods for robustness, security, and resilience of cyberphysical control systems: games-in-games principle for optimal cross-layer resilient control systems," *IEEE Control Systems Magazine*, vol. 35, no. 1, pp. 46–65, 2015.

[24] Y. Shoukry and P. Tabuada, "Event-triggered state observers for sparse sensor noise/attacks," *IEEE Transactions on Automatic Control*, vol. 61, no. 8, pp. 2079–2091, 2015.

[25] R. Ivanov, M. Pajic, and I. Lee, "Attack-resilient sensor fusion for safety-critical cyber-physical systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 15, no. 1, pp. 1–24, 2016.

[26] X. Ding, S. L. Smith, C. Belta, and D. Rus, "Optimal control of Markov decision processes with linear temporal logic constraints," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1244–1257, 2014.

[27] S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ-calculus specifications," in *the Proceedings of the 48th IEEE Conference on Decision and Control (CDC)*, pp. 2222–2229, 2009.

[28] M. Lahijanian, S. B. Andersson, and C. Belta, "Temporal logic motion planning and control with probabilistic satisfaction guarantees," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 396–409, 2012.

[29] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning for dynamical systems," in *the Proceedings of the 48th IEEE Conference on Decision and Control (CDC)*, pp. 5997–6004, 2009.

[30] E. M. Wolff, U. Topcu, and R. M. Murray, "Robust control of uncertain markov decision processes with temporal logic specifications," in *the Proceedings of the 51st IEEE Conference on Decision and Control (CDC)*, pp. 3372–3379, 2012.

[31] N. Vlassis, M. L. Littman, and D. Barber, "On the computational complexity of stochastic controller optimization in POMDPs," *ACM Transactions on Computation Theory*, vol. 4, no. 4, 2012.

[32] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 971–984, 2000.

[33] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[34] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *the Proceedings of International Conference on Robotics and Automation (ICRA)*, pp. 2689–2696, IEEE, 2010.

[35] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Motion planning with dynamics by a synergistic combination of layers of planning," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 469–482, 2010.

[36] D. Sadigh and A. Kapoor, "Safe control under uncertainty with probabilistic signal temporal logic," in *Robotics: Science and Systems*, 2016.

[37] D. Fudenberg and J. Tirole, *Game Theory*. MIT Press, 1991.

[38] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5319–5325, IEEE, 2014.

[39] M. B. Horowitz, E. M. Wolff, and R. M. Murray, "A compositional approach to stochastic optimal control with co-safe temporal logic specifications," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1466–1473, IEEE, 2014.

[40] I. Papusha, J. Fu, U. Topcu, and R. M. Murray, "Automata theory meets approximate dynamic programming: Optimal control with temporal logic constraints," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 434–440, IEEE, 2016.

[41] H. J. Kappen, "Linear theory for control of nonlinear stochastic systems," *Physical review letters*, vol. 95, no. 20, p. 200201, 2005.

[42] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 462–467, 2007.

[43] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[44] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[45] M. Srinivasan and S. Coogan, "Control of mobile robots using barrier functions under temporal logic specifications," *arXiv preprint arXiv:1908.04903*, 2019.

[46] M. Srinivasan, S. Coogan, and M. Egerstedt, "Control of multi-agent systems with finite time control barrier certificates and temporal logic," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 1991–1996, IEEE, 2018.

[47] L. Lindemann and D. V. Dimarogonas, "Control barrier functions for signal temporal logic tasks," *IEEE control systems letters*, vol. 3, no. 1, pp. 96–101, 2018.

[48] G. Yang, C. Belta, and R. Tron, "Continuous-time signal temporal logic planning with control barrier functions," in *2020 American Control Conference (ACC)*, pp. 4612–4618, 2020.

[49] M. Zhu and S. Martinez, "Stackelberg-game analysis of correlated attacks in cyber-physical systems," in *the Proceedings of American Control Conference*, pp. 4063–4068, IEEE, 2011.

[50] Q. Zhu and T. Başar, "Robust and resilient control design for cyber-physical systems with an application to power systems," in *the Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference*, pp. 4066–4071, 2011.

[51] N. Basilico, N. Gatti, and F. Amigoni, "Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder," *Artificial Intelligence*, vol. 184, pp. 78–123, 2012.

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

[53] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive (1) designs," *Journal of Computer and System Sciences*, vol. 78, no. 3, pp. 911–938, 2012.

[54] R. Alur, S. Moarref, and U. Topcu, "Compositional synthesis with parametric reactive controllers," in *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pp. 215–224, ACM, 2016.

[55] A. N. Kulkarni and J. Fu, "A compositional approach to reactive games under temporal logic specifications," in *2018 Annual American Control Conference (ACC)*, pp. 2356–2362, IEEE, 2018.

[56] A. N. Kulkarni and J. Fu, "Opportunistic synthesis in reactive games under information asymmetry," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 5323–5329, IEEE, 2019.

[57] V. Raman and H. Kress-Gazit, "Analyzing unsynthesizable specifications for high-level robot behavior using LTLMoP," in *Computer Aided Verification*, pp. 663–668, Springer, 2011.

[58] V. Raman and H. Kress-Gazit, "Automated feedback for unachievable high-level robot behaviors," in *the Proceedings of International Conference on Robotics and Automation (ICRA)*, pp. 5156–5162, IEEE, 2012.

[59] A. Cimatti, M. Roveri, V. Schuppan, and A. Tchaltsev, "Diagnostic information for realizability," in *International Workshop on Verification, Model Checking, and Abstract Interpretation*, pp. 52–67, Springer, 2008.

[60] J. Tumová, L. I. R. Castro, S. Karaman, E. Frazzoli, and D. Rus, "Minimum-violation LTL planning with conflicting specifications," in *the Proceedings of American Control Conference (ACC)*, pp. 200–205, IEEE, 2013.

[61] J. Tumova, S. Karaman, C. Belta, and D. Rus, "Least-violating planning in road networks from temporal logic specifications," in *the Proceedings of International Conference on Cyber-Physical Systems (ICCPS)*, pp. 1–9, IEEE, 2016.

[62] C.-I. Vasile, J. Tumova, S. Karaman, C. Belta, and D. Rus, "Minimum-violation scLTL motion planning for mobility-on-demand," in *the Proceedings of International Conference on Robotics and Automation (ICRA)*, pp. 1481–1488, IEEE, 2017.

[63] P. Chaudhari, T. Wongpiromsarny, and E. Frazzoli, "Incremental minimum-violation control synthesis for robots interacting with external agents," in *the Proceedings of American Control Conference (ACC)*, pp. 1761–1768, IEEE, 2014.

[64] M. Lahijanian, S. Almagor, D. Fried, L. E. Kavraki, and M. Y. Vardi, "This time the robot settles for a cost: A quantitative approach to temporal logic planning with partial satisfaction.," in *AAAI*, pp. 3664–3671, 2015.

[65] R. Dimitrova, M. Ghasemi, and U. Topcu, "Maximum realizability for linear temporal logic specifications," in *International Symposium on Automated Technology for Verification and Analysis*, pp. 458–475, Springer, 2018.

[66] S. Henning, S. Philipp, and M. Bürger, "On the design of penalty structures for minimum-violation LTL motion planning," in *the Proceedings of International Conference on Decision and Control (CDC)*, IEEE, 2018.

[67] K. Kim, G. Fainekos, and S. Sankaranarayanan, "On the minimal revision problem of specification automata," *The International Journal of Robotics Research*, vol. 34, no. 12, pp. 1515–1535, 2015.

[68] M. Guo and D. V. Dimarogonas, "Reconfiguration in motion planning of single-and multi-agent systems under infeasible local LTL specifications," in *the Proceedings of International Conference on Decision and Control (CDC)*, pp. 2758–2763, IEEE, 2013.

[69] M. Guo, K. H. Johansson, and D. V. Dimarogonas, "Revising motion planning under linear temporal logic specifications in partially known workspaces," in *the Proceedings of International Conference on Robotics and Automation (ICRA)*, pp. 5025–5032, IEEE, 2013.

[70] M. Guo and D. V. Dimarogonas, "Distributed plan reconfiguration via knowledge transfer in multi-agent systems under local LTL specifications," in *the Proceedings of International Conference on Robotics and Automation (ICRA)*, pp. 4304–4309, IEEE, 2014.

[71] F. Buccafurri, T. Eiter, G. Gottlob, and N. Leone, "Enhancing model checking in verification by AI techniques," *Artificial Intelligence*, vol. 112, no. 1, pp. 57–104, 1999.

[72] E. Bartocci, R. Grosu, P. Katsaros, C. Ramakrishnan, and S. Smolka, "Model repair for probabilistic systems," *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 326–340, 2011.

[73] A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien, "Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation," in *International Conference on Intelligent Robots and Systems*, vol. 2, pp. 963–972, IEEE, 1996.

[74] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[75] R. I. Brafman, "A heuristic variable grid solution method for POMDPs," in *AAAI/IAAI*, pp. 727–733, 1997.

[76] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces.," in *Robotics: Science and Systems*, 2008.

[77] R. Sharan and J. Burdick, "Finite state control of POMDPs with LTL specifications," in *Proceedings of the American Control Conference*, pp. 501–508, 2014.

[78] H. Yu and D. P. Bertsekas, "On near optimality of the set of finite-state controllers for average cost POMDP," *Mathematics of Operations Research*, vol. 33, no. 1, pp. 1–11, 2008.

[79] E. A. Hansen, D. S. Bernstein, and S. Zilberstein, "Dynamic programming for partially observable stochastic games," in *AAAI*, vol. 4, pp. 709–715, 2004.

[80] D. S. Bernstein, E. A. Hansen, and S. Zilberstein, "Bounded policy iteration for decentralized POMDPs," in *International Joint Conference on Artificial Intelligence*, pp. 52–57, 2005.

[81] D. Szer and F. Charpillet, "An optimal best-first search algorithm for solving infinite horizon DEC-POMDPs," in *European Conference on Machine Learning*, pp. 389–399, Springer, 2005.

[82] C. Amato, D. S. Bernstein, and S. Zilberstein, "Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs," *Autonomous Agents and Multi-Agent Systems*, vol. 21, no. 3, pp. 293–320, 2010.

[83] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.

[84] S. K. Dhall and C. L. Liu, "On a real-time scheduling problem," *Operations research*, vol. 26, no. 1, pp. 127–140, 1978.

[85] C. A. Phillips, C. Stein, E. Torng, and J. Wein, "Optimal time-critical scheduling via resource augmentation," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pp. 140–149, 1997.

[86] S. Karaman and E. Frazzoli, "Vehicle routing problem with metric temporal logic specifications," in *47th IEEE Conference on Decision and Control*, pp. 3953–3958, IEEE, 2008.

[87] W. S. Jewell, "Markov-renewal programming: Formulation, finite return models," *Operations Research*, vol. 11, no. 6, p. 938, 1963.

[88] M. Pinedo, *Scheduling*, vol. 29. Springer, 2012.

[89] S. M. Ross, *Introduction to Stochastic Dynamic Programming*. Academic Press, 2014.

[90] L. I. Sennott, *Stochastic Dynamic Programming and the Control of Queueing Systems*, vol. 504. John Wiley & Sons, 2009.

[91] H. C. Tijms, *A First Course in Stochastic Models*. John Wiley and sons, 2003.

[92] S. Stidham and R. Weber, "A survey of Markov decision models for control of networks of queues," *Queueing systems*, vol. 13, no. 1-3, pp. 291–314, 1993.

[93] J. Liu and P. Prabhakar, "Switching control of dynamical systems from metric temporal logic specifications," in *IEEE International Conference on Robotics and Automation*, pp. 5333–5338, 2014.

[94] Y. Zhou, D. Maity, and J. S. Baras, "Timed automata approach for motion planning using metric interval temporal logic," in *European Control Conference (ECC)*, pp. 690–695, IEEE, 2016.

[95] D. Beauquier, "On probabilistic timed automata," *Theoretical Computer Science*, vol. 292, no. 1, pp. 65–84, 2003.

[96] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic games for verification of probabilistic timed automata," in *International Conference on Formal Modeling and Analysis of Timed Systems*, pp. 212–227, Springer, 2009.

[97] P. Bouyer and V. Forejt, "Reachability in stochastic timed games," in *International Colloquium on Automata, Languages, and Programming*, pp. 103–114, Springer, 2009.

[98] T. Brázdil, J. Krčál, J. Křetínskỳ, A. Kučera, and V. Řehák, "Stochastic real-time games with qualitative timed automata objectives," in *International Conference on Concurrency Theory*, pp. 207–221, Springer, 2010.

[99] Z. Feng, R. Dearden, N. Meuleau, and R. Washington, "Dynamic programming for structured continuous markov decision problems," *arXiv preprint arXiv:1207.4115*, 2012.

[100] W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov chain Monte Carlo in practice*. CRC Press, 1995.

[101] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. II. Athena Scientific, 2011.

[102] A. Sinha, F. Fang, B. An, C. Kiekintveld, and M. Tambe, "Stackelberg security games: Looking beyond a decade of success," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 2018.

[103] S. G. Loizou and K. J. Kyriakopoulos, "Automatic synthesis of multi-agent motion tasks based on LTL specifications," in *the Proceedings of the 43rd IEEE Conference on Decision and Control (CDC)*, vol. 1, pp. 153–158, 2004.

[104] M. Lahijanian, S. B. Andersson, and C. Belta, "A probabilistic approach for control of a stochastic system from ltl specifications," in *the Proceedings of the 48th IEEE Conference on Decision and Control/Chinese Control Conference*, pp. 2236–2241, 2009.

[105] C. Y. Ma, N. S. Rao, and D. K. Yau, "A game theoretic study of attack and defense in cyber-physical systems," in *the Proceedings of IEEE Conference on Computer Communications Workshops*, pp. 708–713, 2011.

[106] J. v. Neumann, "Zur theorie der gesellschaftsspiele," *Mathematische annalen*, vol. 100, no. 1, pp. 295–320, 1928.

[107] L. Hogben, *Handbook of Linear Algebra*. Chapman and Hall/CRC, 2013.

[108] C. R. Fox and Y. Rottenstreich, "Partition priming in judgment under uncertainty," *Psychological Science*, vol. 14, no. 3, pp. 195–200, 2003.

[109] J. Pita, M. Jain, M. Tambe, F. Ordóñez, and S. Kraus, "Robust solutions to Stackelberg games: Addressing bounded rationality and limited observations in human cognition," *Artificial Intelligence*, vol. 174, no. 15, pp. 1142–1171, 2010.

[110] H. Xu, B. Ford, F. Fang, B. Dilkina, A. Plumptre, M. Tambe, M. Driciru, F. Wanyama, A. Rwetsiba, M. Nsubaga, *et al.*, "Optimal patrol planning for green security games with black-box attackers," in *International Conference on Decision and Game Theory for Security*, pp. 458–477, Springer, 2017.

[111] J. Pita, M. Jain, J. Marecki, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus, "Deployed armor protection: the application of a game theoretic model for security at the Los Angeles International Airport," in *the Proceedings of the 7th International joint Conference on Autonomous agents and Multiagent Systems: Industrial Track*, pp. 125–132, International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[112] A. X. Jiang, Z. Yin, C. Zhang, M. Tambe, and S. Kraus, "Game-theoretic randomization for security patrolling with dynamic execution uncertainty," in *the Proceedings of International Conference on Autonomous Agents and Multi-Agent Systems*, pp. 207–214, International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[113] Y. Vorobeychik and S. P. Singh, "Computing Stackelberg equilibria in discounted stochastic games.," in *AAAI*, 2012.

[114] J. Hu, M. C. Fu, and S. I. Marcus, "A model reference adaptive search method for global optimization," *Operations Research*, vol. 55, no. 3, pp. 549–568, 2007.

[115] T. Homem-de Mello, "A study on the cross-entropy method for rare-event probability estimation," *INFORMS Journal on Computing*, vol. 19, no. 3, pp. 381–394, 2007.

[116] S. Burer and A. N. Letchford, "Non-convex mixed-integer nonlinear programming: A survey," *Surveys in Operations Research and Management Science*, vol. 17, no. 2, pp. 97–106, 2012.

[117] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT Press, 2005.

[118] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, 1972.

[119] H. Royden and P. Fitzpatrick, *Real Analysis*. Prentice Hall, 2010.

[120] V. Conitzer, "On Stackelberg mixed strategies," *Synthese*, vol. 193, pp. 689–703, 2016.

[121] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust optimization*. Princeton University Press, 2009.

[122] P. Poupart and C. Boutilier, "Bounded finite state controllers," in *Neural Information Processing Systems*, pp. 823–830, 2004.

[123] R. Alur, T. Feder, and T. A. Henzinger, "The benefits of relaxing punctuality," *Journal of the ACM*, vol. 43, no. 1, pp. 116–146, 1996.

[124] J. Wang, W. Tu, L. C. K. Hui, S. M. Yiu, and E. K. Wang, "Detecting time synchronization attacks in cyber-physical systems with machine learning techniques," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 2246–2251, June 2017.

[125] F. Pasqualetti, F. Dörfler, and F. Bullo, "Attack detection and identification in cyber-physical systems," *IEEE transactions on automatic control*, vol. 58, no. 11, pp. 2715–2729, 2013.

[126] O. Cappé, S. J. Godsill, and E. Moulines, "An overview of existing methods and recent advances in sequential monte carlo," *Proceedings of the IEEE*, vol. 95, no. 5, pp. 899–924, 2007.

[127] E. A. Hansen, "Solving POMDPs by searching in policy space," in *Conf. on Uncertainty in Artificial Intelligence*, pp. 211–219, 1998.

[128] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical computer science*, vol. 126, no. 2, pp. 183–235, 1994.

[129] S. Coogan, E. A. Gol, M. Arcak, and C. Belta, "Traffic network control from temporal logic specifications," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 2, pp. 162–172, 2015.

[130] B. Yordanov, J. Tumova, I. Cerna, J. Barnat, and C. Belta, "Temporal logic control of discrete-time piecewise affine systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1491–1504, 2012.

[131] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[132] A. Li, L. Wang, P. Pierpaoli, and M. Egerstedt, "Formally correct composition of coordinated behaviors using control barrier certificates," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3723–3729, IEEE, 2018.

[133] T. Wongpiromsarn, U. Topcu, and A. Lamperski, "Automata theory meets barrier certificates: Temporal logic verification of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3344–3355, 2015.

[134] X. Xu, "Constrained control of input–output linearizable systems using control sharing barrier functions," *Automatica*, vol. 87, pp. 195–201, 2018.

[135] W. M. Haddad, S. G. Nersesov, and L. Du, "Finite-time stability for time-varying nonlinear dynamical systems," in *2008 American control conference*, pp. 4135–4139, IEEE, 2008.

[136] T. Babiak, F. Blahoudek, M. Křetínskỳ, and J. Strejček, "Effective translation of ltl to deterministic rabin automata: Beyond the (F, G)-fragment," in *Automated Technology for Verification and Analysis*, pp. 24–39, Springer, 2013.

[137] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*, pp. 6271–6278, IEEE, 2014.