Optimizing Rewards while meeting ω-regular Constraints

Christopher K. Zeitler
ckzeitler@gmail.com
Rational CyPhy Inc., Urbana, IL, USA

Kristina Miller
University of Illinois, Urbana-Champaign, IL, USA

Sayan Mitra
University of Illinois, Urbana-Champaign, IL, USA

John Schierman
Air Force Research Laboratory

Mahesh Viswanathan
University of Illinois, Urbana-Champaign, IL, USA

Abstract

This paper addresses the problem of synthesizing policies for Markov Decision Processes (MDPs) with hard ω-regular constraints, which include and are more general than safety, reachability, liveness, and fairness. The objective is to derive a policy that not only makes the MDP adhere to the given ω-regular constraint T with certainty but also maximizes the expected reward. We first show that there are no optimal policies for the general constrained MDP (CMDP) problem with ω-regular constraints, which contrasts with simpler problem of CMDPs with safety requirements. Next we show that, despite its complexity, the optimal policy can be approximated within any desired level of accuracy in polynomial time. This approximation ensures both the fulfillment of the ω-regular constraint with probability 1 and the attainment of a ϵ-optimal reward for any given $\epsilon > 0$. The proof identifies specific classes of policies capable of achieving these objectives and may be of independent interest. Furthermore, we introduce an approach to tackle the CMDP problem by transforming it into a classical MDP reward optimization problem, thereby enabling the application of conventional reinforcement learning. We show that proximal policy optimization is an effective approach to identifying near-optimal policies that satisfy ω-regular constraints. This result is demonstrated across multiple environments and constraint types.

1 Introduction

Rewards shape an agent's behavior, but finding the right reward is generally a hard problem. For an agent modeled as a Markov Decision Process (MDP), at each step the agent takes an action and collects a reward, and as its goal is maximizing the accumulated reward, the choice of the reward defines its optimal behavior. For real world agents facing different costs, objectives, levels of information, and constraints, the right reward may not be obvious. For example, for a fleet of autonomous wingmen, a natural reward signal will be the distance to the leading aircraft so that they are encouraged to keep-up, but at the same time the wingmen should maintain safe separation with each other and the leader, and there may be other factors like minimizing fuel usage. In practice, the different components are combined into a single reward with coefficients, and the exact coefficients are selected via computationally intensive hyper-parameter tuning.

Adding *constraints* to the agent's behavior can alleviate some of the above challenges. Now the agents behavior is defined by maximizing the accumulated reward, *while meeting* all the specified constraints. For the real world examples, it is often straightforward to enumerate the hard constraints like maintaining separation and reaching targets. Introducing constraints lead to the *Constrained MDP (CMDP)* framework which adds a penalty function $c(s, a)$ for each state s and action a, a constraint $C(s_t) = F(c(s_t, a_t), \ldots, c(s_N, a_N))$, and a threshold $C(s_t) \leq \alpha$ (see the book by (Altman, 2021) for a review). Typically, the function F computes the total, expected, or the discounted penalty over a time horizon, and the constrained optimization problem aims to assure that this never crosses the threshold. With these types of constraints, however, it is still difficult to compute the right threshold that *guarantees* that really an unsafe state will never be visited, or that desired target states will always be visited.

The motivation for using MDPs for safety-critical systems has therefore led to recent works on CMDPs with *hard constraints* and reinforcement learning algorithms for solving them (Yu et al., 2022). Previous works focus on hard safety constraints and incorporate ideas like recoverable sets (Miller et al., 2024), control barrier functions (CBF) (Ames et al., 2019), and safety indices (SI) to solve the constrained optimization problem.

This work studies the problem of CMDPs with hard *ω-regular constraints*. ω-regular constraints include safety, reachability, liveness, and fairness properties, and all properties expressible in temporal logics like LTL. Formally, the problem we consider is the following. Given an MDP M, an ω-regular property T, a reward function r and discount factor γ, the goal is to synthesize a policy that, among those that satisfy T with probability 1, is the one that maximizes the expected reward as per (r, γ). We show that, in general, there are no optimal policies for this problem. It is worth contrasting this observation against the following results: (a) optimal positional policies exist to maximize discounted rewards (Puterman, 1994), (b) optimal finite memory, pure policies exist to maximize the probability of satisfying an ω-regular property (Baier & Katoen, 2008), and (c) optimal positional policies exist that satisfy a hard invariance constraint and optimize a discounted reward (Miller et al., 2024).

Our first result shows that the CMDP problem with hard ω-regular constraints can nonetheless be *approximated in polynomial time*. That is, given any $\epsilon > 0$, in polynomial time we can synthesize a strategy that satisfies a given ω-regular property T with probability 1 *and* earns reward that is within ϵ of the optimal. The proof of this result relies on showing that both the classes of pure, finite memory policies and stationary policies, contain a policy that can satisfy the hard constraint and earn reward that is close to optimal. The existence of special policies may be of independent interest in finding other algorithmic solutions.

Next, we show that the problem of identifying a policy that has close to optimal reward while meeting a hard ω-regular constraint, can be reduced to the classical problem of optimizing discounted rewards (without hard constraints) on a slightly modified MDP. Thus, classical algorithms like reinforcement learning that solve MDP optimization can be brought to bear to solve the new problem. Our reduction once again exploits the observation that pure, finite memory policies can achieve close to optimal rewards while meeting the hard, logic constraint.

We evaluate the effectiveness of a number of reinforcement learning approaches in finding close to optimal policies that satisfy hard ω-regular constraints. We demonstrate the applicability of this approach across a number of discrete control examples wherein the policy consists of a choice between different aircraft controllers, and the continuous control example of optimal orbit transfer of a satellite. The diversity of these examples shows the broad applicability of our reward-shaping approach.

Related Work (Tessler et al., 2019) introduce Reward Constrained Policy Optimization (RCPO) which incorporates constraints as a penalty signal into the reward function and show that this algorithm converges almost surely to a constraint satisfying policy, under mild assumptions.

The CRL formulation does not capture the fact that hard constraints, like safety, are about worst-case behavior and not so much about cumulative or discounted costs. This has been addressed in Reachability constrained RL (RCRL) (Yu et al., 2022), which modifies CRL to impose the safety constraints over the entire time horizon, without discounting. It finds a policy π, that maximizes a cost that combines the usual reward with a penalty for violating safety, with the constraint that for all recoverable states π never violates safety. Furthermore, (Yu et al., 2022) uses the a version of Lagrange multiplier method to solve RCRL, which is a standard for CRL.

Traditionally, policies maximizing the probability of satisfying ω-regular properties have been identified using techniques like linear programming, value iteration, and policy iteration that require knowledge of the entire state space (Baier & Katoen, 2008). More recently, there has been interest in using RL to solve this problem (Sadigh et al., 2014; Hahn et al., 2019; Alur et al., 2021; 2023). By carefully engineering a reward function, these algorithms use RL to synthesize policies that satisfy ω-regular properties with a probability that is close to maximum. Unlike this paper, these results do not consider an additional reward that must be maximized along with satisfying a property.

Theoretical results on synthesizing policies that maximize two reward functions with the same discount factor are presented in (Chatterjee et al., 2006). The problem of maximizing the probability of satisfying multiple ω-regular properties is considered in (Etessami et al., 2008). Both these papers show that policies that approximate the Pareto curve for multi-objective problems can be synthesized in polynomial time by reducing the problem to multi-objective linear programming. In comparison, results in this paper handle the case where one objective that is discounted (i.e., reward) and the other is not (i.e., probability of satisfying the property). This subtle change introduces challenges that we overcome.

The results presented in (Voloshin et al., 2022) are closest in spirit to this paper. The problem they try to solve is following: given a finite state MDP M, an LTL formula φ, and a cost function, find among the policies that satisfy φ with highest probability, the one the optimizes the cost. The cost function in (Voloshin et al., 2022) is a hybrid cost function that combines average cost and transient cost. In contrast, we consider discounted rewards in this paper. Since we require our policy to satisfy the hard constraint with probability 1, modulo the difference in the cost functions considered in the two papers, (Voloshin et al., 2022) look at a more general problem. Taking π^* to denote the policy that optimizes cost among those that maximize the probability of satisfying φ, (Voloshin et al., 2022) present a PAC learning algorithm that constructs (with high probability) a policy that satisfies φ with probability that is close to the probability with which π^* satisfies φ and has cost that is close to the cost that π^* has. Thus, their algorithm does not solve the problem we consider here because, even if there are policies that satisfy the hard constraint with probability 1, the algorithm in (Voloshin et al., 2022) will not necessarily find it. Further, the algorithm in (Voloshin et al., 2022) relies on very strong assumptions about the MDP, unlike the RL-based algorithm presented here.

2 Discounted Reward Markov Decision Processes

Notation. The set of all probability distributions over a finite set S will be denoted by $\mathcal{D}(S)$. For an element $s \in S$, the *dirac distribution* δ_s is the probability distribution where $\delta_s(s) = 1$ and $\delta_s(s') = 0$ for all $s' \neq s$. The *support* of a distribution μ is the set $\text{supp}(\mu) = \{s \in S \mid \mu(s) > 0\}$. A (finite or infinite) sequence/string/word w over a set S is a sequence of the form $w = s_0 s_1 \cdots s_{n-1} \cdots$ where $s_i \in S$ is the ith element of the sequence which we denote by $w(i)$. The *length* of such a word w, denoted $|w|$, is the length of the sequence; thus, $|w| \in \mathbb{N}$ or it maybe infinite. The set of finite of words over S is denoted as S^* and the set of infinite words as S^ω. For non-empty sets A and B, we sometimes consider sequences that alternate between A and B. Thus $(AB)^\omega$ is the set of all infinite sequences where the elements in the even positions are in A, and the elements in the odd positions are in B. Similarly, $(AB)^*A$ are finite sequences with even elements in A, odd elements in B, and the last element in A, while $(AB)^*$ are finite sequences with even elements in A, odd elements in B and last element in B (unless the sequence is empty).

Markov Decision Process (MDP). A Markov Decision Process (MDP) is a tuple $M = (Q, A, \Delta, q_0)$ where Q is a finite set of *states*, A is a finite set of *actions*, $q_0 \in Q$ is the start/initial state, and the (partial) function $\Delta : Q \times A \hookrightarrow \mathcal{D}(Q)$ is the transition function. The set of actions *enabled* in $q \in Q$ is $A(q) = \{a \in A \mid \Delta(q,a) \text{ is defined}\}$. We assume (without loss of generality) that $A(q) \neq \emptyset$ for every state $q \in Q$. A *run* of M is an alternating sequence of states and actions $\rho \in (QA)^\omega$ such that $\rho(0) = q_0$ and for every i, $\rho(2i+1) \in A(\rho(2i))$ and $\rho(2i+2) \in \text{supp}(\Delta(\rho(2i), \rho(2i+1)))$. The set of all runs of M will be denoted as $\mathsf{Runs}(M)$. A *finite run* is $\eta \in (QA)^*Q$ such that there is a run $\rho \in \mathsf{Runs}(M)$ with η as a prefix; the set of all finite runs of M will be denoted as $\mathsf{Runs}_f(M)$.

Markov Chain. A Markov Chain is an MDP $M = (Q, A, \Delta, q_0)$ such that $|A| = 1$. A Markov chain defines a probability measure on $\mathsf{Runs}(M)$ as follows. For any finite run η, the *cylinder* set C_η is $\{\rho \in \mathsf{Runs}(M) \mid \eta \text{ is a prefix of } \rho\}$. The set of measurable sets over $\mathsf{Runs}(M)$ is taken to be the σ-field generated by the collection of all cylinder sets C_η for any $\eta \in \mathsf{Runs}_f(M)$. The probability of C_η is given by $\mu_M(C_\eta) = \prod_{i:\ 2i+2 \leq |\eta|} \Delta(\eta(2i), a)(\eta(2i+2))$ where a is the unique action of M. The probability over the σ-field is the unique measure that extends the above function on cylinder sets; we denote that by μ_M as well. Many natural subsets of runs are measurable, including those defined by temporal logics like LTL. Given a subset $S \subseteq Q$, the following collections of runs are measurable.

Safety $\Box S = \{\rho \in \mathsf{Runs}(M) \mid \forall i.\ \rho(2i) \in S\}$, i.e., runs where every state is in S

Reachability $\Diamond S = \{\rho \in \mathsf{Runs}(M) \mid \exists i.\ \rho(2i) \in S\}$, i.e., runs where some state is in S.

Fairness $\Box\Diamond S = \{\rho \in \mathsf{Runs}(M) \mid \forall i. \exists j > i.\ \rho(2j) \in S\}$, i.e., runs where infinitely many states are in S.

As we will describe later, every ω-regular property can be recast as a fairness property for an appropriate Markov Chain, and it is a generalization of both safety and reachability. Finally, given a random variable X on $\mathsf{Runs}(M)$, we use $\mathbb{E}_M[X]$ to denote the expectation of X with respect to the distribution μ_M.

Policies. Let $M = (Q, A, \Delta, q_0)$ be an MDP. A policy resolves the non-deterministic choices in an MDP. Formally, a *policy* (for M) is a function $\sigma : \mathsf{Runs}_f(M) \to \mathcal{D}(A)$ such that for any finite run η, $\text{supp}(\sigma(\eta)) \subseteq A(\eta(|\eta|))$. Thus, a policy maps a finite run to a distribution on next actions whose support is restricted to those that are enabled at the last state of η. An MDP M together with a policy σ, induces a Markov chain $M^\sigma = (\mathsf{Runs}_f(M), \{a\}, \Delta^\sigma, q_0)$ where $a \notin A$ is the unique action of M^σ and

$$\Delta^\sigma(\eta, a)(\eta b q) = \sigma(\eta)(b)\Delta(\eta(|\eta|), b)(q).$$

Informally the states of M^σ are finite runs of M, and the probability of transitioning from a run η to a run $\eta b q$ is given by the probability that σ chooses b at η times the probability of transitioning to state q from the last state of η on action b. The Markov chain M^σ has countably many states. [1]

A policy σ is said to be *deterministic* if $|\text{supp}(\sigma(\eta))| = 1$ for every finite run η, i.e., σ chooses a single action with probability 1 from every finite run. If a policy is not deterministic, we will say it is *randomized*. A policy σ is *stationary* if the choice of action only depends on the last state, i.e., for every $\eta_1, \eta_2 \in \mathsf{Runs}_f(M)$, if $\eta_1(|\eta_1|) = \eta_2(|\eta_2|)$ then $\sigma(\eta_1) = \sigma(\eta_2)$. The last type of policies we consider are *finite memory* policies, where the decision on the next action is made based on a finite amount of information stored about the run. To define it formally, let us consider a Myhill-Nerode type congruence, where we will say for $\eta_1, \eta_2 \in \mathsf{Runs}_f(M)$, $\eta_1 \equiv_\sigma \eta_2$ if for every $\kappa \in (AQ)^*$, $\sigma(\eta_1\kappa) = \sigma(\eta_2\kappa)$. Now a policy σ is *finite memory* if the equivalence $\equiv_\sigma$ has finitely many equivalence classes. We conclude by observing that if the policy is stationary or finite memory, the Markov chain M^σ is equivalent (w.r.t. to bisimulation) to a finite state Markov chain. It is useful to explicitly define this "equivalent finite state" Markov chain when σ is a stationary policy on M; we will abuse notation and also call this M^σ. Formally, when σ is stationary, for a MDP $M = (Q, A, \Delta, q_0)$, $M^\sigma = (Q, \{a\}, \Delta^\sigma, q_0)$ where $a \notin A$ is the unique action of the Markov chain and

$$\Delta^\sigma(q, a)(p) = \sum_{b \in A} \sigma(q)(b)\Delta(q, b)(p).$$

[1]While the Markov chains in this paper will almost always have finitely many states (and finitely many actions), there will be rare occasions when we consider ones with countably many states (but finitely many actions).

Measures of Sets. Given a measureable set $T \subseteq \mathsf{Runs}(M)$ of an MDP M, one classical problem is to find a policy σ that maximizes the probability of the set T. For the properties considered in this paper, it is well known that deterministic, stationary policies are sufficient to maximize the probability of these properties. Moreover, the deterministic and stationary policy that maximizes the probability, works no matter what we take to be the initial state.

Proposition 2.1 (Lemma 10.102 and Exercise 10.23 of (Baier & Katoen, 2008))**.** *Let $M = (Q, A, \Delta, q_0)$ be an MDP and $S \subseteq Q$, a subset of states. Let $T \subseteq \mathsf{Runs}(M)$ be one of $\Box S$, $\Diamond S$ or $\Box\Diamond S$. For $q \in Q$, let $M_q = (Q, A, \Delta, q)$ be the MDP M with initial state q. There is a deterministic and stationary policy σ_* such that for any $q \in Q$ $\mu_{M_q^{\sigma_*}}(T) = \sup_\sigma \mu_{M_q^\sigma}(T)$.*

Discounted Rewards. Rewards model features that we would like policies to satisfy. A *reward structure* for an MDP $M = (Q, A, \Delta, q_0)$ is a pair (r, γ), where $r : (Q \times A) \to \mathbb{R}$ is the *reward function* and $\gamma \in (0, 1)$ (open interval between 0 and 1) is the *discount factor*. A reward structure (r, γ) defines a random variable $X_{(r,\gamma)}$ on the runs of M that assigns a reward to every run as follows.

$$X_{(r,\gamma)}(\rho) = \sum_{i \in \mathbb{N}} \gamma^i r(\rho(2i), \rho(2i+1)).$$

The discount factor ensures that the above infinite sum converges. One is usually interested in finding a policy for an MDP that maximizes the expected value of the random variable defined by the reward structure. A classical observation about discounted rewards is that for a finite state MDP, there is a deterministic and stationary policy that maximizes expected reward. Like in Proposition 2.1, the same positional policy works for all initial states.

Proposition 2.2 (Theorem 6.2.7 of (Puterman, 1994))**.** *Let $M = (Q, A, \Delta, q_0)$ be an MDP and (r, γ) a reward structure. For $q \in Q$, let $M_q = (Q, A, \Delta, q)$ be the MDP M with initial state q. There is a deterministic and stationary policy σ_* such that for any $q \in Q$ $\mathbb{E}_{M_q^{\sigma_*}}[X_{(r,\gamma)}] = \sup_\sigma \mathbb{E}_{M_q^\sigma}[X_{(r,\gamma)}]$.*

Next, it is known that the reward achieved by any policy σ can also be achieved by a stationary (not necessarily deterministic) policy. Notice that this is a very different statement than Proposition 2.2.

Theorem 2.3 (Theorem 5.5.3 of (Puterman, 1994))**.** *For any MDP M, policy σ, and discount factor $\gamma \in (0, 1)$, there is a stationary (not necessarily deterministic) policy σ_* such that for any reward function r,*

$$\mathbb{E}_{M^{\sigma_*}}[X_{(r,\gamma)}] = \mathbb{E}_{M^\sigma}[X_{(r,\gamma)}].$$

Optimizing Rewards with Hard Constraints. The problem we consider in this paper is to optimize rewards while meeting hard constraints. Given an MDP M, a rewards structure (r, γ), and a measureable set of runs $T \subseteq \mathsf{Runs}(M)$, the goal is to find a policy σ such that it satisfies T with probability 1 while maximizing the expected reward. We define the problem as finding the maximum reward (while satisfying T) instead of computing an optimal policy, since, as we shall see, the "optimal" policies may not exist.

Constrained MDP Optimization Problem. Given an MDP M, a reward structure (r, γ), and a measurable set of runs $T \subseteq \mathsf{Runs}(M)$ compute $\sup_{\sigma:\mu_{M^\sigma}(T)=1} \mathbb{E}_{M^\sigma}[X_{(r,\gamma)}]$.

A recent result (Miller et al., 2024) shows that when $T = \Box S$ for some subset of states S, optimal policies exist and they can be deterministic and stationary. Furthermore, there is an effective algorithm to find this optimal policy through reward shaping.

Theorem 2.4 (Miller et al. (2024))**.** *Let M be an MDP and S a subset of states such that there is a policy σ' such that $\mu_{M^{\sigma'}}(\Box S) = 1$. Then for any reward structure (r, γ), there is reward function r' such that*

$$\sup_\sigma \mathbb{E}_{M^\sigma}[X_{(r',\gamma)}] = \sup_{\sigma:\mu_{M^\sigma}(\Box S)=1} \mathbb{E}_{M^\sigma}[X_{(r,\gamma)}].$$

Additionally there is a deterministic and stationary policy σ_* such that $\mu_{M^{\sigma_*}}(\Box S) = 1$ and

$$\mathbb{E}_{M^{\sigma_*}}[X_{(r',\gamma)}] = \mathbb{E}_{M^{\sigma_*}}[X_{(r,\gamma)}] = \sup_{\sigma:\mu_{M^\sigma}(\Box S)=1} \mathbb{E}_{M^\sigma}[X_{(r,\gamma)}].$$

Theorem 2.4 provides an algorithm to solve the Constrained MDP Optimization Problem for safety properties — construct the new reward structure (r', γ), and search for the optimal positional policy by ignoring the hard safety constraint using any one of the many algorithms for this problem like linear programming or reinforcement learning.

Unfortunately, Theorem 2.4 doesn't extend to the other properties we consider in this paper, namely reachability or fairness. Not only can we not guarantee the existence of an optimal positional policy, there may in fact be *no* optimal policy. This can be seen through Example A.1 in the Appendix.

3 Polynomial Time Approximation Algorithm

Constrained MDP Optimization Problem may not have optimal solutions when the hard constraint is reachability or fairness (Example A.1). The best one can do in such a scenario is to find policies that are arbitrarily close to optimal, i.e., given $\epsilon > 0$, find a policy that satisfies the hard constraint and gets expected reward that is within ϵ of the optimal possible reward. The main result of this section establishes that this problem is in *polynomial time.* This result applies not only to reachability and fairness but (unsurprisingly) to all ω-regular hard constraints. Our proof of this result relies on obtaining something analogous to Propositions 2.1 and 2.2, that "special" policies suffice to approximate the optimal reward while meeting the hard constraint. Of course, as Example A.1 shows, these special policies *cannot* be both deterministic and stationary. Instead we show that the next best thing possible holds: we show that both the classes of stationary (but not necessarily deterministic) and deterministic, finite memory policies are sufficient.

Our first result shows that deterministic, finite memory policies can approximate the optimal reward while meeting a hard constraint.

Theorem 3.1. *Let $M = (Q, A, \Delta, q_0)$ be an MDP, $S \subseteq Q$ a subset of states, and (r, γ) a reward structure. Suppose there is a policy σ' such that $\mu_{M^{\sigma'}}(\Box\Diamond S) = 1$. For any $\epsilon > 0$, there is a deterministic, finite memory policy σ_* such that $\mu_{M^{\sigma_*}}(\Box\Diamond S) = 1$ and*

$$\mathbb{E}_{M^{\sigma_*}}[X_{(r,\gamma)}] \geq \left(\sup_{\sigma : \mu_{M^\sigma}(\Box\Diamond S)=1} \mathbb{E}_{M^\sigma}[X_{(r,\gamma)}] \right) - \epsilon.$$

Proof Sketch. Taking $M_q = (Q, A, \Delta, q)$ to be the MDP M with initial state q, define $Q_1 = \{q \in Q \mid \sup_\sigma \mu_{M_q^\sigma}(\Box\Diamond S) = 1\}$. From Proposition 2.1, we can conclude that these are the states from which $\Box\Diamond S$ can be satisfied with probability 1. Let M_1 be the MDP restricted to states in Q_1, i.e., $M_1 = (Q_1, A, \Delta_1, q_0)$ where $\Delta_1(q, a)$ is defined and equal to $\Delta(q, a)$ if $\Delta(q, a)$ is defined and $\text{supp}(\Delta(q, a)) \subseteq Q_1$. Let σ_r be the deterministic and stationary policy that maximizes the reward due to (r, γ) in M_1 as guaranteed by Proposition 2.2, i.e., $\mathbb{E}_{M_1^{\sigma_r}}[X_{(r,\gamma)}] = \sup_\sigma \mathbb{E}_{M_1^\sigma}[X_{(r,\gamma)}]$. Next, let σ_g be the deterministic and stationary policy that satisfies $\Box\Diamond S$ with probability 1 in M_1 as guaranteed by Proposition 2.1. That is, taking $M_{1,q} = (Q_1, A, \Delta_1, q)$ to be the MDP M_1 with initial state q, $\mu_{M_{1,q}^{\sigma_g}}(T) = 1$ for all $q \in Q_1$. For $k \in \mathbb{N}$, let σ_k be the policy that follows σ_r for the first k-steps, and from the $k + 1$st step onwards follows σ_g. σ_k is deterministic (since both σ_r and σ_g are) and is finite memory since it only needs to count to k to decide which of σ_r and σ_g to follow. Observe that σ_k stays within Q_1, and $\mu_{M^{\sigma_k}}(\Box\Diamond S) = 1$ as σ_g ensures that $\Box\Diamond S$ is satisfied with probability 1 from any state in Q_1. Finally, one can show that by choosing a sufficiently large k, σ_k approximates the optimal reward to within ϵ. Details for why this holds can be found in the full proof that is in Appendix B.2. $\square$

Next, we can show that stationary policies are also rich enough to solve the Constrained MDP Optimization Problem approximately.

Theorem 3.2. *Let $M = (Q, A, \Delta, q_0)$ be an MDP, $S \subseteq Q$ a subset of states, and (r, γ) a reward structure. Suppose there is a policy σ' such that $\mu_{M^{\sigma'}}(\Box\Diamond S) = 1$. For any $\epsilon > 0$, there is a stationary* *policy* σ_* *such that* $\mu_{M^{\sigma_*}}(\Box\Diamond S) = 1$ *and*

$$\mathbb{E}_{M^{\sigma_*}}[X_{(r,\gamma)}] \geq \left(\sup_{\sigma:\mu_{M^\sigma}(\Box\Diamond S)=1} \mathbb{E}_{M^\sigma}[X_{(r,\gamma)}] \right) - \epsilon.$$

Proof Sketch. Let σ_f be the pure, finite memory policy that approximates the optimal reward as guaranteed by Theorem 3.1. The desired stationary policy that proves this theorem is the stationary policy σ_* corresponding to σ_f guaranteed by Theorem 2.3. The challenge in completing this proof is to argue that σ_* satisfies $\Box\Diamond S$ with probability 1. Details are in Appendix B.2. $\Box$

Theorems 3.1 and 3.2 allow us to prove that we can solve the Constrained MDP Optimization Problem approximately in polynomial time.

Theorem 3.3. *Given an MDP* M*, a property* $T = \Box\Diamond S$ *where* S *is a subset of states, a reward structure* (r, γ)*, and* $\epsilon > 0$ *there is a polynomial time algorithm that either outputs "no policy" if there is no policy* σ *such that* $\mu_{M^\sigma}(T) = 1$ *or outputs a policy* σ_* *such that* $\mu_{M^{\sigma_*}}(T) = 1$ *and*

$$\mathbb{E}_{M^{\sigma_*}}[X_{(r,\gamma)}] \geq \left(\sup_{\sigma:\mu_{M^\sigma}(T)=1} \mathbb{E}_{M^\sigma}[X_{(r,\gamma)}] \right) - \epsilon.$$

Proof is postponed to Appendix B.3.

ω-regular Properties. ω-regular properties are properties of infinite executions that can be recognized by finite state machines — infinite sequence analogues of the classical regular languages. They are very general and include not only the properties we consider in this paper (namely, safety, reachability, and fairness) but also those expressed using temporal logics like LTL. Our observations on solving the Constrained MDP Optimization Problem for fairness constraints, allow us to conclude that the same results hold when the hard constraint is an ω-regular constraint. The solution lies in *reducing* the ω-regular case to the case of fairness. ω-regular properties are typically described using finite automata with infinitary acceptance conditions like Büchi, Rabin, parity, etc. (Baier & Katoen, 2008). To determine the probability of satisfying an ω-regular property T in an MDP M, the idea is to take the cross product of an automaton representation of T with M, and compute the measure of paths that visit certain states infinitely often in the product MDP, i.e., checking the type of fairness constraint we consider in this paper. For this approach to work, the automata representing T must be *limit deterministic Büchi* (Courcoubetis & Yannakakis, 1995). An algorithm to convert an automaton for a language into a limit deterministic Büchi automaton for the same language can be found in (Courcoubetis & Yannakakis, 1995). Efficient translations of temporal logics to limit deterministic automata can be found in (Kini & Viswanathan, 2017a;b).

4 Reward Shaping

The results in Section 3 show that the Constrained MDP Optimization Problem can be approximated in polynomial time. The algorithm presented in Theorem 3.3 relies on having a full description of the MDP. In this section, we will present an alternate approach for solving the Constrained MDP Optimization Problem. We will show that given an instance of Constrained MDP Optimization Problem, namely, an MDP M, reward structure (r, γ), a ω-regular property T, and $\epsilon > 0$, we can find another reward function r' such that finding a policy that optimizes (r', γ), identifies a policy that approximately solves the Constrained MDP Optimization Problem instance. The reduction of approximating Constrained MDP Optimization Problem to classical MDP optimization not only provides another argument for why the problem is in polynomial, but additionally suggests bringing to bear other approaches that do not suffer from the challenges of the algorithm outlined in Theorem 3.3. For example, classical reinforcement learning can be used to solve Constrained MDP Optimization Problem.

Let us fix an MDP $M = (Q, A, \Delta, q_0)$, and a reward structure (r, γ). For a fairness property $\Box\Diamond S$, $S \subseteq Q$, our reduction will rely on assessing a penalty if the policy does not reach certain target states within a certain number of steps. Thus, we need to work with MDP M where in addition, we count the number of steps that have been taken. Second, in our modified MDP after a certain number of steps, whenever we reach a state in S, we will transition to our target set of states with some small probability, and with the remaining probability continue as before. These intuitions are captured in the following definition of a new MDP derived from M. For $o < m \in \mathbb{N}$, $\xi \in (0, 1)$, and $S \subseteq Q$, the *m-unfolding* of M for *property $\Box\Diamond S$ with cut-off o* is the MDP $M[m, o, \xi, \Box\Diamond S] = (Q_1, A, \Delta_1, (q_0, 0))$ where $Q_1 = Q \times \{i \in \mathbb{N} \,|\, i \leq m+2\}$ and Δ_1 is defined to capture the following intuition. $M[m, o, \xi, \Box\Diamond S]$ has $m+2$ copies of the states of M. The number of steps are counted by advancing from a state in one level to a state in the next level. Step counting stops once the MDP reaches level $m+1$ or $m+2$. Starting from level o, each time a state in S is visited, with probability ξ it moves to level $m+2$ regardless of what the current level is, and with probability $1-\xi$, moves to the next level. Formally this is defined as follows. First, $\Delta_1((q, i), a)$ is defined iff $\Delta(q, a)$ is defined, and when defined it is

$$\Delta_1((q,i),a)(p,j) = \begin{cases} \Delta(q,a)(p) & \text{if } j = i+1 \leq o \text{ or } i = j = m+2 \\ \Delta(q,a)(p) & \text{if } q \notin S \text{ and either } o < j = i+1 \leq m+1 \text{ or } i = j = m+1 \\ \xi\Delta(q,a)(p) & \text{if } q \in S,\ o \leq i \leq m+1 \text{ and } j = m+2 \\ (1-\xi)\Delta(q,a)(p) & \text{if } q \in S \text{ and either } o < j = i+1 \leq m+1 \text{ or } i = j = m+1 \\ 0 & \text{otherwise} \end{cases}$$

The MDPs M and $M[m, o, \xi, \Box\Diamond S]$ are equivalent in the formal sense of bisimulation. They also have the same set of policies, though they have different state spaces; we will abuse notation and use the same name for a policy for M and the equivalent one for $M[m, o, \xi, \Box\Diamond S]$. We identify a special class of policies for $M[m, o, \xi, \Box\Diamond S]$ that we will focus on. A policy σ is *k-memory policy* for $M[m, o, \xi, \Box\Diamond S]$ (for $k < o$) if it is positional and for every $i, j \geq k+1$, $\sigma(q, i) = \sigma(q, j)$. In other words, the positional strategy σ does not distinguish between two copies of state q in $M[m, o, \xi, \Box\Diamond S]$ when the counter value is $\geq k+1$.

Theorem 4.1. *Let $M = (Q, A, \Delta, q_0)$ be an MDP, (r, γ) a reward structure for M, and $S \subseteq Q$ a subset of states. For $\epsilon > 0$, let $v_\epsilon = \sup_{\sigma:\mu_{M^\sigma}(\Box\Diamond S)=1} \mathbb{E}_{M^\sigma}[X_{(r,\gamma)}] - \epsilon$ [2]. For any $\epsilon > 0$, there are $k < o < m \in \mathbb{N}$, $\xi \in \mathbb{R}$, threshold $\tau \in \mathbb{R}$, and reward function r' on $M[m, o, \xi, \Box\Diamond S]$ such that the following property holds. Let σ_* be a k-memory policy that is optimal with respect to (r', γ) among k-memory policies. That is, $\mathbb{E}_{M[m,o,\xi,\Box\Diamond S]^{\sigma_*}}[X_{(r',\gamma)}] = \sup_{\sigma:k\text{-memory}} \mathbb{E}_{M[m,o,\xi,\Box\Diamond S]^{\sigma}}[X_{(r',\gamma)}]$ [3]. Then $\mathbb{E}_{M^{\sigma_*}}[X_{(r,\gamma)}] \geq v_\epsilon$. Further, there is a policy σ such that $\mu_{M^\sigma}(\Box\Diamond S) = 1$ if and only if $\mathbb{E}_{M[m,o,\xi,\Box\Diamond S]^{\sigma_*}}[X_{(r',\gamma)}] \geq \tau$.*

Thus Theorem 4.1 says that finding an optimal k-memory policy for $M[m, o, \xi, \Box\Diamond S]$ (with appropriate parameters) and reward structure (r', γ), determines if there is a policy σ such that $\mu_{M^\sigma}(\Box\Diamond S) = 1$ and ϵ-approximately solves the Constrained MDP Optimization Problem. The proof of Theorem 4.1 is in Appendix B.4.

5 Experimental Evaluation

In this section, we aim to answer the following research questions through experiments:

- Can our algorithm optimize rewards while meeting hard ω-regular constraints (specifications) via reinforcement learning?
- Which reinforcement learning methods are most effective at reaching optimal policies for hard ω-regular specifications?

[2]As always, $\sup_{a \in A} f(a)$ is taken to be $-\infty$ if $A = \emptyset$.

[3]It is easy to see that such a policy σ_* exists since the supremum is really a maximum as the number of k-memory policies is finite.

- Does our algorithm perform better than other methods with respect to optimizing reward and minimizing constraint violations?

Benchmarks. To answer these questions, we tested our approach in four benchmarking scenarios. The first three scenarios extend the work of (Miller et al., 2024), adding ω-regular constraints to runtime assurance tasks. The fourth benchmark is an optimal orbit transfer task for satellites. Each benchmark is described by an *agent model*, an *environment*, a *reward structure*, and a target ω-regular constraint.

Aircraft Scenarios. In the first three scenarios (Figure 1), the agent is an aircraft (or in the Fleet scenario, a group of two aircrafts) moving in 3D-space with Dubins dynamics. The agent tracks a leader aircraft through the use of two controllers, S and U. The action space for the agent is discrete, consisting of the selection of which controller (or combination of controllers in the Fleet example) to use at a given time step. For the Dubins scenario, the observation space has nine values [4] consisting of the displacement and velocity difference between the agent and the leader, as well as a counter of the time elapsed in the scenario. For the Dubins+O scenario, the observation space is expanded to 27 variables to additionally encode the displacement to and spacial extent of the obstacles. In the Fleet scenario, the observation space is an extension of the Dubins observation space to include the displacement and velocity difference between all three aircraft. While it is atypical to include the episode time in the observation space because this causes the model to lose invariance to time translation, note that the ω-regular constraints being imposed are time dependent (e.g., reaching a certain location within a certain time limit). Therefore, it is critical that the models *not* be time invariant so that the constraints can be met. This can also be seen as a way of allowing the learning of the k parameter from Theorem 3.1. The initial reward structure r_U aims to encourage the use of the experimental controller U and provides a reward of 1 during time steps that U is used, and zero otherwise:

$$r_U^i(q, a_i) = \begin{cases} 1 & \text{if } a_i = U \\ 0 & \text{otherwise.} \end{cases}$$

As these scenarios extend previous scenarios that ensured safety via reward shaping, we refer to r_U shaped for safety as r_s, while r_ω refers to r_U shaped for both safety and ω-regular constraint satisfaction. Both r_s and r_ω depend on the shaped reward value r': $r' = \frac{-r_{max}}{(1-\gamma)(\gamma^T)}$, where γ is the discount factor, r_{max} is the maximum possible reward for one time step, and T is the maximum episode length. Then, if the safety constraint is violated or if the ω-regular constraint is not met at time T, a reward of r' is applied.

In the Dubins example, the safety constraint is that the agent must always be more than 100 units from the leader, and the reachability constraint is that the agent reaches a target region relative to the leader. In the Dubins+O example, the agent must reach a target region relative to the leader after navigating past two obstacles while maintaining safe separation with the obstacles and the leader. Finally, in the Fleet scenario, the two aircraft must both reach a target region infinitely often while maintaining separation with each other and the leader (See Fig. 1 for diagrams of these scenarios). This is challenging because if both aircrafts attempt to enter the target region, the safety constraint will be violated, and so a successful policy must balance the concerns of fair reaching and safety. In order to enforce such a hard constraint in a fixed-length episode, we implement the following scheme. We track whether each aircraft has entered the target region. Once both aircraft have entered the region (this need not happen simultaneously), the episode transitions to a success state with probability p. With probability $1-p$, the episode continues with each aircraft's flags for entering the reach region reset. In this way, the agents are incentivized to fairly enter the reach region as often as possible to maximize the chance of transitioning to a success state during the episode. All these examples are constructed to put the ω-regular constraint in tension with the reward structure; simply maximizing r_U leads the property to not be fulfilled (See baseline results in Table 1).

[4] While it may appear that this only requires seven values, all vectors are encoded as a magnitude and a unit vector, and so use four values instead of the necessary three.

Optimal Orbit Transfer. The final satellite example is based on autonomous spaceship operations (Jewison & Erwin, 2016; Johnson et al., 2012). In this scenario, the agent is a satellite with two-dimensional Clohessy–Wiltshire dynamics

$$\ddot{x} = 3n^2 x + 2n\dot{y} + u_x$$
$$\ddot{y} = -2n\dot{x} + u_y,$$

where n is the orbital rate. This agent has a continuous action space that determines the amount of thrust u to apply on a given time step. The ω-regular constraint is that the satellite must transit to within a distance δ of a target orbit.

We measure this distance in the space of natural motion trajectory (NMT) parameters (Ichimura & Ichikawa, 2008). For example, if the satellite is on an orbit characterized by the NMT parameters a_s, d_s, c_s and the goal orbit has parameters a_g, d_g, and c_g, then the satellite has achieved the constraint if $\left\| \left[a_s - a_g, \quad d_s - d_g, \quad c_s - c_g \right] \right\| < \delta$. For scenarios with an obstacle satellite, the agent satellite is required to maintain a distance greater than five units from the obstacle at all times. The base reward r_t consists of a penalty equal to the amount of thrust used on a given time step: $r_t(q, a_i) = -a_i$. The final reward structure r_ω is created in the same fashion as in the aircraft scenarios, combining the base reward r_t with the penalties for not achieving the constraint, or for violating safety, if applicable.

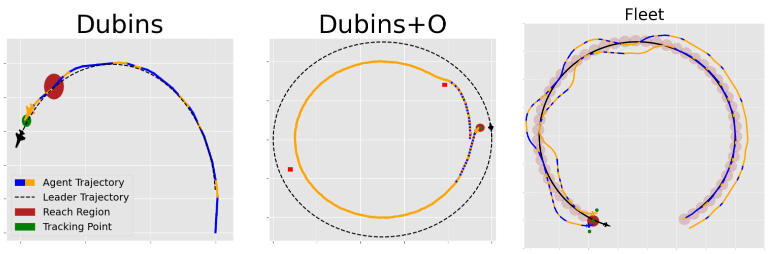

Figure 1: *Left.* A typical Dubins episode. The agent (orange) tracks a point behind the leader (black) and must utilise S to avoid colliding with the leader and to slow down sufficiently to enter the target region (dark red). *Middle.* A typical Dubins+O episode. The agent must switch between S and U to avoid collision with the obstacles (bright red) and to enter the target region between the points that the two controllers track. *Right.* A typical Fleet episode. Both agents must fairly share access to the target region (red) but not simultaneously. This characteristic behavior can be observed in the recent half of their trajectories as the aircraft enter and exit the region in a coordinated fashion. Light red circles show previous locations of the target to emphasize this behavior.

This problem has a known theoretical optimum (Ichimura & Ichikawa, 2008), and so our results can be compared to this value to judge the degree to which our approach is able to achieve optimally. Their result also demonstrates that the optimum can be reached with thrusts only along one axis, and so for simplicity our agent is limited to thrusts along the y-axis. Because of the continuous action space, it is necessary to apply a slightly more complex reward shaping to ensure the agent reaches the desired orbit. Specifically, the shaped reward r_ω for failing to fulfill the ω-regular property is scaled according to the distance to the desired orbit:

$$r_\omega = \begin{cases} 0 & \text{if } d < \delta \\ r' \frac{d}{\delta} & otherwise, \end{cases}$$

where d is the distance to the orbit calculated as described earlier, and r' is the reward shaping value used in the aircraft examples. This modification helps to move the agent towards fulfillment of the ω-regular constraint, because unlike in the discrete case, it is extremely unlikely that the agent fulfills the constraint through random exploration.

Learning Algorithms. We evaluate the benchmarks with four reinforcement learning algorithms.

- *Baseline:* Proximal policy optimization (PPO) (Schulman et al., 2017) without ω-regular reward shaping, using r_s. This provides a point of comparison in the absence of constraints and demonstrates the degree to which the constraint-aware techniques (below) modify the agents behavior.
- *PPO-shaped:* PPO applied to the shaped reward structure r_ω.
- *SAC-shaped:* Soft actor-critic (Haarnoja et al., 2018) applied to r_ω. This strategy helps us investigate the effectiveness of an offline algorithm at this task.
- *CPO:* Constrained policy optimization (CPO) (Achiam et al., 2017), another approach to ensuring hard constraints are not violated by policies.

Notably, CPO attempts to minimize constraint violations during training, while our approach only attempts to reach a final policy that satisfies constraints with no restrictions on training behavior. For CPO, the ω-regular properties are encoded in the algorithm's cost function, and the rewards are unshaped. For these examples, the cost function c_ω is given by

$$c_\omega = \begin{cases} 1 & \text{if constraint is violated} \\ 0 & \text{otherwise.} \end{cases}$$

The algorithms were applied to the benchmark scenarios on an M1 MacBook Pro. The PPO, PPO-shaped, and SAC-shaped examples were trained using Ray's rllib. The CPO benchmark was trained using a publicly available PyTorch implementation of CPO (Sikchi, 2021) with slight modifications to allow for GPU acceleration and discrete action spaces. The hyperparameters used for training can be found in the appendix.

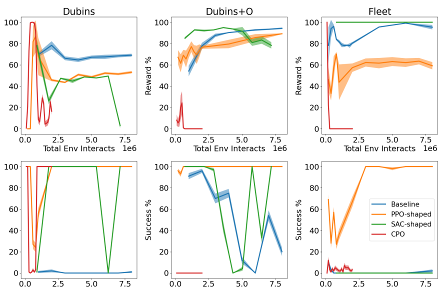

Figure 2: Typical training runs for the aircraft scenarios. Here, the reward r_U is normalized to the episode length, so a reward of 1 indicates that U was used at every time step, and a reward of 0 indicates that S was used at every time step. The results are averaged over 100 randomly-sampled initial conditions with bars representing one standard deviation of sampling error.

Results and observations from Aircraft benchmarks. Fig. 2 displays typical training runs for the four algorithms across the first three scenarios. The policies were trained for approximately eight million environment interactions, except for CPO, which was trained for about 2 million interactions. This is due to the additional computation time costs of the more complex algorithm. Despite this reduced training time, the CPO policy reached a steady-state in two of the three experiments, and its reward was no longer increasing in the third.

On the simplest Dubins task, PPO-shaped, SAC-shaped, and CPO all reached policies that met the hard ω-regular constraints. On the Dubins+O task, PPO-shaped and SAC-shaped attained policies that satisfied constraints, while CPO did not. For the Fleet scenario, only PPO-shaped achieved a policy that consistently satisfied the constraint. Interestingly, the CPO policy converged to a strategy of sending both aircraft into the target region, allowing it to succeed a fraction of the time, and violate the safety constraint otherwise. The SAC-shaped policy was the opposite, never sending the aircraft towards the target, and thus never satisfying the reachability constraint. The SAC-shaped policy converged very early in training, possibly indicating insufficient exploration. Table 1 contains a summary of the best policies for each algorithm across the three scenarios. The baseline policy was selected by choosing the policy with the highest reward, while the other policies were selected by choosing the policy with the highest success rate. We define the success rate (S%) to be the percentage of episodes that the ω-regular constraint was satisfied without violating safety. The reward percent (R%) is defined as the ratio of the final reward to the number of time steps in an episode, because the maximum possible reward on any time step is one. If there were multiple policies with the same highest success rate, then the policy with the highest reward among them was selected. These results indicate that PPO-shaped is the most robust method among those tested for obtaining policies for satisfying ω-regular constraints as it was the only approach that was able to meet the desired requirements across all experiments. In addition, it achieved rewards comparable to or better than other algorithms when those algorithms generated constraint-satisfying policies.

	Dubins		Dubins+O		Fleet	
	R%	S%	R%	S%	R%	S%
Base	78.4±3.8	2.0	94.3±0.7	20.0	99.2±0.3	0
PPO	53.1±1.2	100	89.2±0.4	100	59.4±3.0	100
SAC	48.7±0.3	100	92.8±0.9	100	100±0	0
CPO	40.5±3.4	100	24.3±11	0	22.6±6.6	12

Table 1: A summary of the results from the best policies achieved by each algorithm on the aircraft benchmarks. For the baseline, the policy with the highest reward (U%) was selected. For the other policies, the policy that succeeded at the task most often (S%) was selected. If there were multiple policies with the same S% value, then the policy with the highest U% value among those was selected.

Results and observations from Satellite benchmarks. We trained policies for optimal orbital transfer in two scenarios: one in which the agent is the only object in the environment, and one in which there is an obstacle satellite between the agent's starting location and the goal orbit. In these examples, the agent must transfer between the NMT$(10, 5, 0)$ to the NMT$(50, 10, 0)$. In the scenario with an obstacle satellite, it has an orbit NMT$(25, 0, 0)$. Fig. 3 displays typical training runs using PPO-shaped for both scenarios.

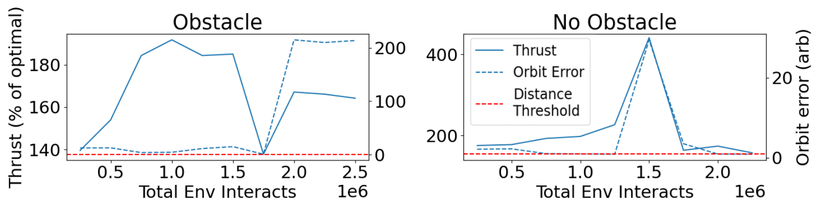

Figure 3: Typical orbit transfer training runs for scenarios with and without an obstacle satellite. These training runs have significant variance, with both the thrust reward and orbit error changing significantly on some updates.

Despite the challenge of the problem, as evidenced by the number of policies that did not satisfy the ω-regular constraint, in both scenarios a policy was reached that completed the desired orbit transfer successfully with energy usages comparable to theoretically optimal values.

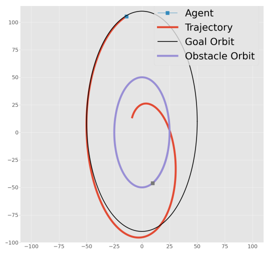

Figure 4: A typical orbit transfer episode with an obstacle (blue). The agent (red) slightly overshoots the desired orbit (black) before correcting back towards it, increasing energy usage.

Fig. 4 displays the trajectory of the best policy attained for the version of the scenario with an obstacle. Compared to the theoretically optimal transfer, this policy used 37% more energy in the orbit transfer. In contrast, the best policy trained without an interfering obstacle used 57% more energy than the optimal transfer. This indicates that the presence obstacles does not impact our method's ability to find efficient transfer strategies, and that this approach to calculating transfers is viable when the complexity of safety constrains makes direct computation of an optimal trajectory infeasible. However, the fact that the simpler scenario did not converge to policy with thrust usage equal to or better than the scenario with an obstacle indicates that there is significant variance in the training process. These training runs display a high degree of variability, with both the thrust and distance error varying significantly between some policy checkpoints. This is due to the finely-tuned nature of the orbit transfer problem; if the thrust do not precisely balance, then the satellite can enter a drifting orbit, moving far from the goal and requiring large energy expenditures to return to the goal orbit.

This variance could also be due to the large mismatch in scale between the two reward components. While this mismatch is also present in the aircraft examples, the magnitude of difference is larger in this example because $\frac{d}{\delta}$ can be large even when only small thrusts are used, leading the reward optimization to be dominated by fulfillment of the ω-regular property, with low sensitivity to optimization of the thrust usage. This is may also be the reason that neither policy achieves the theoretic optimum.

Acknowledgements

The work was supported in part by SBIR Phase II contract FA239124CB023 and NSF SHF 2007428.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization, 2017.

Eitan Altman. *Constrained Markov decision processes.* Routledge, 2021.

Rajeev Alur, Suguman Bansal, Osbert Bastani, and Kishor Jothimurugan. A framework for transforming specifications in reinforcement learning. *Springer Festschrift in honor of Prof. Tom Henzinger*, October 2021. doi: 10.48550/arXiv.2111.00272. URL https://arxiv.org/abs/2111.00272v3.

Rajeev Alur, Osbert Bastani, Kishor Jothimurugan, Mateo Perez, Fabio Somenzi, and Ashutosh Trivedi. Policy synthesis and reinforcement learning for discounted ltl. In *Proceedings of the International Conference on Computer Aided Verification*, 2023.

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pp. 3420–3431. IEEE, 2019.

Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking.* The MIT Press, 2008. ISBN 026202649X, 9780262026499.

Krishnendu Chatterjee, Rupak Majumdar, and Thomas A. Henzinger. Markov decision processes with multiple objectives. In *Proceedings of the Annual Symposium on Theoretical Aspects of Computer Science*, volume 3884 of *Lecture Notes in Computer Science*, pp. 325–336, 2006.

Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.

Kousha Etessami, Marta Kwiatkowska, Moshe Y. Vardi, and Mihalis Yannakakis. Multi-Objective Model Checking of Markov Decision Processes. *Logical Methods in Computer Science*, 4(4), 2008.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Omega-regular objectives in model-free reinforcement learning. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 395–412, 2019.

Yoshihiro Ichimura and Akira Ichikawa. Optimal impulsive relative orbit transfer along a circular orbit. *Journal of Guidance Control and Dynamics*, 31, 07 2008. doi: 10.2514/1.32820.

Christopher Jewison and R Scott Erwin. A spacecraft benchmark problem for hybrid control and estimation. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 3300–3305. IEEE, 2016.

Taylor T. Johnson, Jeremy Green, Sayan Mitra, Rachel F. Dudley, and Richard Scott Erwin. Satellite rendezvous and conjunction avoidance: Case studies in verification of nonlinear hybrid systems. In *FM 2012: Formal Methods - 18th International Symposium, Paris, France, August 27-31, 2012. Proceedings*, pp. 252–266, 2012.

Dileep Kini and Mahesh Viswanathan. Optimal translation of LTL to limit deterministic automata. In *Proceedings of the Internation Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 113–129, 2017a.

Dileep Kini and Mahesh Viswanathan. Complexity of model checking MDPs against LTL specifications. In *Proceedings of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, pp. 35:1–35:13, 2017b.

Kristina Miller, Chris Zeitler, William Shen, Kerianne Hobbs, Sayan Mitra, John Schierman, and Mahesh Viswanathan. Optimal runtime assurance via reinforcement learning. In *Proceedings of the ACM/IEEE International Conference on Cyber Physical Systems*, 2024.

M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, New York, 1994.

D. Sadigh, E. Kim, S. Coogan, S.S. Sastry, and S.A. Seshia. A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications. In *Proceedings of the IEEE Conference on Decision and Control*, pp. 1091–1096, 2014.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Harshit Sikchi. pytorch-CPO, 2021. URL https://github.com/hari-sikchi/pytorch_CPO.

Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkfrvsA9FX.

Cameron Voloshin, Hoang Minh Le, Swarat Chaudhuri, and Yisong Yue. Policy optimization with linear temporal logic constraints. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems*, 2022.

Dongjie Yu, Haitong Ma, Shengbo Li, and Jianyu Chen. Reachability constrained reinforcement learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 25636–25655. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/yu22d.html.

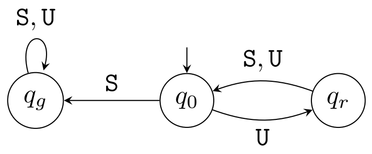

Figure 5: No optimal policy to optimize rewards for hard reachability constraints.

A No optimal policy

Example A.1. MDP $M = (\{q_0, q_r, q_g\}, \{\mathtt{U}, \mathtt{S}\}, \Delta, q_0)$ is shown in Figure 5. From the initial state q_0, U goes to q_r with probability 1, while S goes to q_g with probability 1. Both actions U, S go to q_0 with probability 1 from q_r. Finally, from q_g all actions (U, S) stay in q_g with probability 1. Let $T = \Box\Diamond\{q_g\}$ (which in this case is the same as $\Diamond\{q_g\}$) and $r(q, a) = 1$ if $q = q_r$ and $r(q, a) = 0$ if $q \neq q_r$. Consider the policy σ_k that chooses U for the first $2k - 1$ steps, then chooses S from then on. Observe that $\mu_{M^{\sigma_k}}(T) = 1$ and so σ_k satisfies the hard constraint. Moreover, $\mathbb{E}_{M^{\sigma_k}}[X_{(r,\gamma)}] = \frac{\gamma(1-\gamma^{2k})}{1-\gamma^2}$.

Next, the policy σ_* that chooses U at every step achieves a reward of $\frac{\gamma}{1-\gamma^2}$ and this is the maximum possible reward one can get since it gets a reward of 1 in every odd step. However the policy σ_* does not satisfy the hard constraint since $\mu_{M^{\sigma_*}}(T) = 0$.

Given the observations about σ_k and σ_*, we can say that for any k

$$\frac{\gamma(1-\gamma^{2k})}{1-\gamma^2} \leq \sup_{\sigma:\mu_{M^\sigma}(T)=1} \mathbb{E}_M[X_{(r,\gamma)}] \leq \frac{\gamma}{1-\gamma^2}.$$

Thus, $\sup_{\sigma:\mu_M(T)=1} \mathbb{E}_M[X_{(r,\gamma)}] = \frac{\gamma}{1-\gamma^2}$. However, there is no policy σ such that $\mu_{M^\sigma}(T) = 1$ and $\mathbb{E}_M[X_{(r,\gamma)}] = \frac{\gamma}{1-\gamma^2}$. This is because no reward is received after reaching q_g. Thus, there is no optimal policy amongst those that satisfy T with probability 1. However, we can get arbitrarily close to the optimal reward using policy σ_k by increasing k.

Notice that σ_k is a finite memory, pure policy. How well do deterministic, stationary policies do in this example? Any deterministic, stationary policy is forced to choose between U and S from q_0. The only way to ensure visiting q_g is to choose S. Any such policy gets a reward of 0. Thus, for any deterministic, stationary policy σ_p with $\mu_{M^{\sigma_p}}(T) = 1$, we have $\mathbb{E}_{M^{\sigma_p}}[X_{(r,\gamma)}] = 0$

B Proofs of results from Sections 3 and 4

Before presenting the proofs, we start with some preliminaries that are needed in our proofs.

B.1 Preliminaries

We begin with a proof sketch for Theorem 2.3.

Proof Sketch of Theorem 2.3. Let σ be an arbitrary policy of M. Consider the following reward functions r_q and $r_{(q,a)}$ for each state q and action a defined as follows.

$$r_q(p,b) = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{otherwise} \end{cases}$$
$$r_{(q,a)}(p,b) = \begin{cases} 1 & \text{if } p = q \text{ and } b = a \\ 0 & \text{otherwise} \end{cases}$$

Define the following stationary policy σ_* where

$$\sigma_*(q)(a) = \frac{\mathbb{E}_{M^\sigma}[X_{(r_{(q,a)},\gamma)}]}{\mathbb{E}_{M^\sigma}[X_{(r_q,\gamma)}]}.$$ [5]

One can prove that for any reward function r, $\mathbb{E}_{M^{\sigma_*}}[X_{(r,\gamma)}] = \mathbb{E}_{M^\sigma}[X_{(r,\gamma)}]$. □

Next, we recall some classical observations about the structure of MDPs.

End Components. Let us fix an MDP $M = (Q, A, \Delta, q_0)$. A *sub-MDP* of M is a pair (E, α) such that $E \subseteq Q$ and $\alpha : E \to 2^A$ such that (a) for all $q \in E$, $\emptyset \neq \alpha(q) \subseteq A(q)$, i.e., $\alpha(q)$ is a non-empty subset of the actions enabled at q, (b) for every $q \in E$ and $a \in \alpha(q)$, $\text{supp}(\Delta(q,a)) \subseteq E$. The *underlying graph* of a sub-MDP (E, α), $G_{(E,\alpha)}$, has vertex set $E \cup \{(q,a) \in E \times A \mid a \in \alpha(q)\}$ and edges $(q,(q,a))$ ($q \in E$ and $a \in \alpha(q)$) and $((q,a),p)$ where $p \in \text{supp}(\Delta(q,a))$. A sub-MDP (E,α) of M is an *end component* if the underlying graph $G_{(E,\alpha)}$ is strongly connected. Finally, an end component (E,α) is *maximal* if (E,α) is "maximal" with respect to subset inclusion, i.e., for any other end component (E',α'), if $E \subseteq E'$ and $\alpha(q) \subseteq \alpha'(q)$ for every $q \in E$, then $E = E'$ and $\alpha = \alpha'$. Maximal end components are *disjoint* i.e., if (E_1,α_1) and (E_2,α_2) are maximal end components then $E_1 \cap E_2 = \emptyset$.

For finite MDPs with respect to any policy, runs *almost surely* reach one of the end components.

Proposition B.1 (Theorem 10.120 of (Baier & Katoen, 2008))**.** *Let M be an MDP and σ be a policy for M. Let $T = \Diamond(\cup_{(E,\alpha):\ end\ component} E)$. Then $\mu_{M^\sigma}(T) = 1$.*

The probability of satisfying a fairness property is related to the probability of reaching certain end components. Before proving this, we need a technical definition. For a *stationary* policy σ, an end component (E,α) is *consistent* with σ if for every $q \in E$, $\alpha(q) = \text{supp}(\sigma(q))$. For consistent end components, we can drop α as it is completely determined by E and σ.

Proposition B.2 (Theorems 10.25 and 10.122 of (Baier & Katoen, 2008))**.** *Let M be an MDP, S a set of states, and σ a policy for M. Let*

$$V_{\Box\Diamond S} = \bigcup_{\substack{(E,\alpha):M\ end\ comp.\\ and\ S\cap E\neq\emptyset}} E,$$

be the states that belong to end components in M that contain some state in S. Then, $\mu_{M^\sigma}(\Box\Diamond S) \leq \mu_{M^\sigma}(\Diamond V_{\Box\Diamond S})$.

If σ is a stationary policy, then $\mu_{M^\sigma}(T) = \mu_{M^\sigma}(\Diamond C_{\Box\Diamond S})$ where

$$C_{\Box\Diamond S} = \bigcup_{\substack{(E,\alpha):\sigma\ cons.\ end\ comp.\\ and\ S\cap E\neq\emptyset}} E.$$

B.2 Sufficiency of Special Policies

In this Section, we present the proofs showing that deterministic, finite memory policies and stationary policies can approximate the optimal reward while meeting a hard constraint.

Proof of Theorem 3.1. We can assume without loss of generality that $r(q,a) \geq 0$ for all $q \in Q$ and $a \in A$; if $r(q,a) < 0$ for some $q \in Q$ and A, then we can consider the reward function r_1, where $r_1(q,a) = r(q,a) - \min_{q' \in Q, a' \in A} r(q',a')$ and a policy that optimizes r_1 also optimizes r.

Let us fix $T = \Box\Diamond S$. Let Q_1 be the subset of Q consisting of states from which the property T can be satisfied with probability 1. Taking $M_q = (Q, A, \Delta, q)$ to be the MDP M with initial state q, define

[5] If $\mathbb{E}_{M^\sigma}[X_{(r_q,\gamma)}] = 0$ then q can be removed as q is not reachable. Thus, we can, without loss of generality, assume that $\mathbb{E}_{M^\sigma}[X_{(r_q,\gamma)}] \neq 0$.

$Q_1 = \{q \in Q \mid \sup_\sigma \mu_{M_q^\sigma}(T) = 1\}$. Given the assumption that there is a policy σ' of M that satisfies T with probability 1, we can conclude that $q_0 \in Q_1 \neq \emptyset$. Let M_1 be the MDP restricted to states in Q_1, i.e., $M_1 = (Q_1, A, \Delta_1, q_0)$ where $\Delta_1(q,a)$ is defined and equal to $\Delta(q,a)$ if $\Delta(q,a)$ is defined and $\mathrm{supp}(\Delta(q,a)) \subseteq Q_1$. Let σ_r be the deterministic, stationary policy that maximizes the reward due to (r,γ) in M_1 as guaranteed by Proposition 2.2, i.e., $\mathbb{E}_{M_1^{\sigma_r}}[X_{(r,\gamma)}] = \sup_\sigma \mathbb{E}_{M_1^\sigma}[X_{(r,\gamma)}]$. Next, let σ_g be the deterministic, stationary policy that satisfies T with probability 1 in M_1 as guaranteed by Proposition 2.1. That is, taking $M_{1,q} = (Q_1, A, \Delta_1, q)$ to be the MDP M_1 with initial state q, $\mu_{M_{1,q}^{\sigma_g}}(T) = 1$ for all $q \in Q_1$.

Our first observation is that the reward earned by policy σ_r upper bounds the reward earned by any policy σ' of M that satisfies the constraint T. Let σ' be an arbitrary policy such that $\mu_{M^{\sigma'}}(T) = 1$. Now one can show executions with respect to σ' almost surely stay within Q_1. In other words, $\mu_{M^{\sigma'}}(\Diamond(Q \setminus Q_1)) = 0$. This observation follows from Propositions B.1 and B.2 as $\mu_{M^{\sigma'}}(T) \leq \mu_{M^{\sigma'}}(\Diamond V_T) \leq 1 - \mu_{M^{\sigma'}}(\Diamond(Q \setminus Q_1))$. Thus, σ' is a policy in M_1. Since σ_r maximizes the reward in M_1, we have $\mathbb{E}_{M^{\sigma'}}[X_{(r,\gamma)}] \leq \mathbb{E}_{M^{\sigma_r}}[X_{(r,\gamma)}]$. Therefore,

$$\sup_{\sigma : \mu_{M^\sigma}(T)=1} \mathbb{E}_{M^\sigma}[X_{(r,\gamma)}] \leq \mathbb{E}_{M^{\sigma_r}}[X_{(r,\gamma)}].$$

For $k \in \mathbb{N}$, let σ_k be the policy that follows σ_r for the first k-steps, and from the $k+1$st step onwards follows σ_g. σ_k is deterministic (since both σ_r and σ_g are) and is finite memory since it only needs to count to k to decide which of σ_r and σ_g to follow. Observe that σ_k stays within Q_1, and $\mu_{M^{\sigma_k}}(T) = 1$ as σ_g ensures that T is satisfied with probability 1 from any state in Q_1. We will now show that the reward earned by policy σ_k approaches the reward earned by σ_r as k increases. Let us fix an ordering of the states in Q. Let P be the stochastic matrix corresponding to σ_r, i.e., the (i,j)th entry of P is $\Delta(q_i, \sigma_r(q_i))(q_j)$, where q_i and q_j are the ith and jth states, respectively. Let δ_{q_0} be the column vector representing the Dirac distribution on q_0 and r be the column vector where the ith entry is $r(q_i, \sigma_r(q_i))$ (i.e., the reward earned from q_i when the action is picked according to σ_r). Finally let $r_{\mathsf{max}} = \max_{q \in Q,\ a \in A} r(q,a)$. Then, since $r(q,a) \geq 0$ for all $q \in Q$ and $a \in A$,

$$\begin{aligned}
\mathbb{E}_{M^{\sigma_k}}[X_{(r,\gamma)}] &\geq \mathbb{E}_{M^{\sigma_r}}[X_{(r,\gamma)}] - \left(\sum_{i \geq k+1} \gamma^i \delta_{q_0}^T P^i r\right) \\
&\geq \mathbb{E}_{M^{\sigma_r}}[X_{(r,\gamma)}] - \left(\sum_{i \geq k+1} \gamma^i r_{\mathsf{max}}\right) \\
&\geq \mathbb{E}_{M^{\sigma_r}}[X_{(r,\gamma)}] - \frac{r_{\mathsf{max}} \gamma^{k+1}}{1-\gamma}.
\end{aligned}$$

Taking k such that $\epsilon < \frac{r_{\mathsf{max}}\gamma^{k+1}}{1-\gamma}$, we have

$$\mathbb{E}_{M^{\sigma_k}}[X_{(r,\gamma)}] \geq \mathbb{E}_{M^{\sigma_r}}[X_{(r,\gamma)}] - \epsilon \geq \left(\sup_{\sigma : \mu_{M^\sigma}(T)=1} \mathbb{E}_{M^\sigma}[X_{(r,\gamma)}]\right) - \epsilon.$$

Thus σ_k is the deterministic, finite memory policy that establishes the theorem. □

Next, we have the proof of Theorem 3.2.

Proof of Theorem 3.2. Let σ_f be the deterministic, finite memory policy that approximates the optimal reward as guaranteed by Theorem 3.1. From the proof of Theorem 3.1, $\sigma_f = \sigma_k$ for some k (depending on ϵ) where σ_k uses a deterministic, stationary policy for the first k steps, and switches to another deterministic, stationary policy from then on. We will exploit this structure in completing the proof of this theorem. Let us fix $T = \Box\Diamond S$.

For the proof, it will be useful to consider an MDP that is closely related to M. The *k-unfolding* of M, denoted M_k, is M equipped with a counter that counts the first $k+1$ transitions and stops counting once the counter reaches $k+1$. Formally, $M_k = (Q_k, A, \Delta_k, (q_0, 0))$, where $Q_k = Q \times \{i \in \mathbb{N} | i \leq k+1\}$ and Δ_k of M_k is defined as follows. $\Delta_k((q,i),a)$ is defined iff $\Delta(q,a)$ is defined, and when defined it is

$$\Delta_k((q,i),a)(p,j) = \begin{cases} \Delta(q,a)(p) & \text{if } j = i+1 \leq k+1 \\ & \text{ or } i = j = k+1 \\ 0 & \text{otherwise} \end{cases}$$

Given the definition of Δ_k, one can conclude that if (E,α) is an end component in M_k then $E \subseteq Q \times \{k+1\}$.

Policies over M and M_k are the same. Let $\pi : Q_k \to Q$ be the *projection function* defined as $\pi(q,i) = q$. Define $S_k = \pi^{-1}(S) = \{(q,i) \in Q_k \mid q \in S\}$. For any policy σ, observe that the probability of satisfying $\Box\Diamond S$ in M is the same as the probability of satisfying $\Box\Diamond S_k$ in M_k. As a consequence, we will abuse notation and use T to refer to $\Box\Diamond S$ over M and $\Box\Diamond S_k$ over M_k. Now the finite memory, deterministic policy $\sigma_f = \sigma_k$ is a *deterministic, stationary* policy on M_k with the property that $\mu_{M_k^{\sigma_k}}(T) = \mu_{M^{\sigma_k}}(T) = 1$ and

$$\mathbb{E}_{M^{\sigma_k}}[X_{(r,\gamma)}] \geq \left(\sup_{\sigma : \mu_{M^\sigma}(T)=1} \mathbb{E}_{M^\sigma}[X_{(r,\gamma)}] \right) - \epsilon.$$

Let σ_* be the stationary policy corresponding to σ_k as guaranteed by Theorem 2.3. For σ_*, we know that

$$\mathbb{E}_{M^{\sigma_*}}[X_{(r,\gamma)}] = \mathbb{E}_{M^{\sigma_k}}[X_{(r,\gamma)}] \geq \left(\sup_{\sigma : \mu_{M^\sigma}(T)=1} \mathbb{E}_{M^\sigma}[X_{(r,\gamma)}] \right) - \epsilon.$$

and for any $B \subseteq Q$, $\mu_{M^{\sigma_*}}(\Diamond B) > 0$ iff $\mu_{M^{\sigma_k}}(\Diamond B) > 0$. Thus the reward under policy σ_* is close to optimal. In order to complete the proof we need to show that $\mu_{M^{\sigma_*}}(T) = 1$.

Suppose (for contradiction), $\mu_{M^{\sigma_*}}(T) < 1$. Since $\mu_{M^{\sigma_*}}(T) < 1$, from Propositions B.1 and B.2, we can conclude that there is an end component (E,α) consistent with σ_*, such that $E \cap S = \emptyset$ and $\mu_{M^{\sigma_*}}(\Diamond E) > 0$. Therefore, $\mu_{M^{\sigma_k}}(\Diamond E) = \mu_{M_k^{\sigma_k}}(\Diamond \pi^{-1}(E)) > 0$. From the proof of Theorem 2.3 which sketches the construction of σ_*, the fact that σ_k is a combination of deterministic, stationary policies, and the fact that (E,α) is consistent with σ_*, we can conclude that $a \in \alpha(q)$ for any $q \in E$ iff for some i, $\sigma_k(q,i)(a) = 1$, i.e., a is chosen from q at some step in σ_k. This together with the fact that (E,α) is an end component of M means that once a run in M^{σ_k} reaches E, it never leaves E. Thus, any end component (E',α') consistent with σ_k in M_k and contained in $\pi^{-1}(E)$, is disjoint from S. Let

$$C_E = \bigcup_{\substack{(E',\alpha'): \text{ end comp. cons. with } \sigma_* \\ \text{in } M_k \text{ and } E' \subseteq \pi^{-1}(E)}} E'.$$

Based on the observations above, we can conclude that $\mu_{M_k^{\sigma_k}}(\Diamond C_E) > 0$. Together with Proposition B.2, this means that $\mu_{M^{\sigma_k}}(T) = \mu_{M_k^{\sigma_k}}(T) \leq 1 - \mu_{M_k^{\sigma_k}}(\Diamond C_E) < 1$. This gives us the desired contradiction. Hence, $\mu_{M^{\sigma_*}}(T) = 1$, which establishes the theorem. □

B.3 Approximating Constrained MDP Optimization Problem in Polynomial Time

We present the polynomial time algorithm to approximately solve Constrained MDP Optimization Problem.

Proof of Theorem 3.3. We will show that both the deterministic, finite memory strategy guaranteed by Theorem 3.1 and the stationary policy promised by Theorem 3.2, can be constructed in polynomial time, if they exist.

The algorithm will first construct the MDP M_1 over states Q_1 consisting of all states of M from which there is a policy such that T can be satisfied with probability. M_1 can be constructed in polynomial time using for example the algorithm outlined in the proof of Theorem 10.127 in (Baier & Katoen, 2008). If Q_1 does not contain the initial state, then the algorithm answers "no policy" as required. If the initial state belongs to Q_1, then the policies σ_r and σ_g as in the proof of Theorem 3.1 can be computed in polynomial time using standard algorithms like linear programming. The desired finite memory policy runs σ_r for some number of steps and then σ_g; the number of steps k is approximately $\log \epsilon + \log \gamma - 1 - \log r_{\mathsf{max}}$ (see proof of Theorem 3.1) where r_{max} is the maximum reward one can get in any step. This completes the description of polynomial time algorithm to output a finite memory, deterministic policy.

A stationary policy to the solve the problem can be computed by first computing a deterministic, finite memory policy, and then outputting the stationary policy corresponding to it as given by Theorem 2.3. This again can be done in polynomial time. $\square$

B.4 Reward Shaping

In this section, we prove our reward shaping result. Before doing so, we introduce some notation. There is a natural projection mapping $\pi : Q_1 \to Q$ from the states of $M[m, o, \xi, \Box\Diamond S]$ to the states of M which is defined as $\pi(q, i) = q$. Its inverse is defined usual: for a subset $B \subseteq Q$, we define $\pi^{-1}(B) = \{(q, i) \in Q_k \mid q \in B\}$.

Proof of Theorem 4.1. As argued in the proof of Theorem 3.1, we will assume without loss of generality that $r(q, a) \geq 0$ for all $q \in Q$ and $a \in A$. Let us fix ϵ. Let $n = |Q|$ be the number of states in M. Recall that from the proof of Theorem 3.1, there is a finite memory, deterministic policy built from two deterministic, stationary policies, where the first policy is used for the first k-steps and the second policy is used from the $k + 1$st step onwards, which $\epsilon/2$-approximates the Constrained MDP Optimization Problem. Let k be this value given by the proof of Theorem 3.1 for $\epsilon/2$. Let us take $o = k + n$ and $m = k + \ell n$ for some $\ell > 1$ to be set later. We will also fix the exact value of ξ later in the proof. Let ρ be the least non-zero probability of path of length n in M; thus, if one from state q_1 to q_2 in M within n steps, then the probability of this path is at least ρ. Similarly, let η be the least non-zero probability path of length k in M. Next, let $r_{\max} = \max_{(q,a)\in Q\times A} r(q, a)$ be the maximum reward in any step under r.

Let us now define the new reward function r' as follows.

$$r'((q, i), a) = \begin{cases} r(q, a) & \text{if } i \neq m \\ -p & \text{if } i = m \end{cases}$$

Here $p \in \mathbb{R}_{>0}$ should be thought of as a "penalty" and will be set later in this proof. Intuitively, r' assigns the same reward as r except for the $m + 1$st transition which is assigned p if level $m + 2$ is not reached by then. On the other hand, if level $m + 2$ is reached before the $m + 1$st transition, you get the same reward as r.

We have now specified all the pieces and we will establish some properties that will build towards proving the theorem. From now on we will restrict our attention to k-memory policies. We will call a k-memory policy σ *fair* if $\mu_{M[m,o,\xi,\Box\Diamond S]^\sigma}(\Box\Diamond\pi^{-1}(S)) = \mu_{M^\sigma}(\Box\Diamond S) = 1$; otherwise, we will call it *unfair*. Let us define $L_{m+2} = \{(q, m + 2) \mid q \in Q\}$ to be the states in level $m + 2$ in $M[m, o, \xi, \Box\Diamond S]$.

Unfair policies have a low probability of reaching L_{m+2} within m steps. Let σ be an unfair k-memory policy. Since $\mu_{M[m,o,\xi,\Box\Diamond S]^\sigma}(\Box\Diamond\pi^{-1}(S)) < 1$, from Propositions B.1 and B.2, we can conclude that there is an end component (E, α) of $M[m, o, \xi, \Box\Diamond S]$ consistent with σ, such that $E \cap \pi^{-1}(S) = \emptyset$ and $\mu_{M[m,o,\xi,\Box\Diamond S]^\sigma}(\Diamond E) > 0$. Since E is an end component, once a run reaches E, it never leaves E. Further, based on the definition of $M[m, o, \xi, \Box\Diamond S]$, there are no transitions from E to L_{m+2}. Finally, since σ is a k-memory strategy, if $\mu_{M[m,o,\xi,\Box\Diamond S]^\sigma}(\Diamond E) > 0$, E

is reached with non-zero probability within $k+n$ steps. Based on our assumptions, this means that $\mu_{M[m,o,\xi,\Box\Diamond S]^\sigma}(\Diamond E) > \eta\rho > 0$. Consequently, $\mu_{M[m,o,\xi,\Box\Diamond S]^\sigma}(\Diamond L_{m+2}) < 1-\eta\rho$.

Fair policies have a high probability of reaching L_{m+2} within m steps. Let σ be a fair k-memory policy. Since σ is k-memory, the policy chooses the same action from every copy of any state $q \in Q$ after k steps. Further, since $\mu_{M[m,o,\xi,\Box\Diamond S]^\sigma}(\Box\Diamond\pi^{-1}(S)) = 1$, from Propositions B.1 and B.2, we can conclude that from every state (q,i) for $i > k$ reached, there is a non-zero probability of reaching a state in $\pi^{-1}(S)$ within n-steps. Thus, after k-steps, in every trajectory we have a probability of at least $\rho\xi$ of reaching a state in L_{m+2} within n steps. Therefore, after $m = k+\ell n$-steps, the probability of not reaching L_{m+2} is at most $(1-\rho\xi)^\ell$. Hence, the probability of reach L_{m+2} within $m+1$ steps in the Markov chain $M[m,o,\xi,\Box\Diamond S]^\sigma$ is $\geq 1-(1-\rho\xi)^\ell$, which can be made as high as we want by increasing ℓ.

Before proving our next set of observations, it is worth noting that r' is the same as r, except for the penalty it assesses from states in level m. Thus the reward earned by a policy σ in M with respect to (r,γ) in a trajectory is the same as that earned by σ in $M[m,o,\xi,\Box\Diamond S]$ with respect to (r',γ) except on those trajectories that end up in a state in level m after m-steps.

Unfair policies earn low reward with respect to (r',γ). Let σ be an unfair k-memory policy. We know that $\mu_{M[m,o,\xi,\Box\Diamond S]^\sigma}(\Diamond L_{m+2}) < 1-\eta\rho$ and so the probability of not being in a state in L_{m+2} after m-steps is at least $\eta\rho$. As observed, these are the trajectories that get a penalty on the $m+1$st step under r'. Given that the maximum reward possible with respect to (r,γ) is $r_{\max}/(1-\gamma)$, we get that

$$\mathbb{E}_{M[m,o,\xi,\Box\Diamond S]^\sigma}[X_{(r',\gamma)}] < \frac{r_{\max}}{1-\gamma} - \gamma^m p(\eta\rho).$$

This can be made as small as we want by setting p to be large number.

Reward earned by fair policies is similar in (r,γ) and (r',γ). Let σ be a fair policy. As observed earlier, the probability of not reaching L_{m+2} within m-steps is at most $(1-\rho\xi)^\ell$, and again these are the only trajectories that earn a different reward under r'. Thus, we can say that

$$\mathbb{E}_{M[m,o,\xi,\Box\Diamond S]^\sigma}[X_{(r',\gamma)}] \geq \mathbb{E}_{M^\sigma}[X_{(r,\gamma)}] - \gamma^m p(1-\rho\xi)^\ell$$

which can be made as small as we want by increasing ℓ, no matter what we set p and ξ to be. Since we assumed that $r(q,a) \geq 0$ for all $(q,a) \in Q \times A$, we know that $\mathbb{E}_{M^\sigma}[X_{(r,\gamma)}] \geq 0$. Thus, taking threshold $\tau = -\gamma^m p(1-\rho\xi)^\ell$, we can see that $\mathbb{E}_{M[m,o,\xi,\Box\Diamond S]^\sigma}[X_{(r',\gamma)}] \geq \tau$, which establishes the last part of the theorem.

Based on all these observations, to satisfy the theorem, we can set the values of ξ, p, ℓ so that the following inequalities hold.

$$\tau = -\gamma^m p(1-\rho\xi)^\ell > \frac{r_{\max}}{1-\gamma} - \gamma^m p(\eta\rho)$$
$$-\tau = \gamma^m p(1-\rho\xi)^\ell < \epsilon/2$$

Let us now complete the proof of the theorem using our observations and the values of the parameters. Let σ_* be the optimal k-memory policy for $M[m,o,\xi,\Box\Diamond S]$ with respect to (r',γ). Observe that if there is no policy σ such that $\mu_{M^\sigma}(\Box\Diamond S) = 1$, then $v_\epsilon = -\infty$ and the claim about $\mathbb{E}_{M^{\sigma_*}}[X_{(r,\gamma)}] \geq v_\epsilon$ holds trivially.

Let us consider the case when there is a policy σ such that $\mu_{M^\sigma}(\Box\Diamond S) = 1$. We know that by Theorem 3.1 and our choice of k, there is a k-memory policy σ_f such that $\mu_{M^{\sigma_f}}(\Box\Diamond S) = 1$ and

$\mathbb{E}_{M^{\sigma_f}}[X_{(r,\gamma)}] \geq v_\epsilon + \epsilon/2$. Based on our choice of τ and our previous observations, we have

$$\begin{aligned}
\mathbb{E}_{M^{\sigma_*}}[X_{(r,\gamma)}] &\geq \mathbb{E}_{M[m,o,\xi,\Box\Diamond S]^{\sigma_*}}[X_{(r',\gamma)}] \\
&\geq \mathbb{E}_{M[m,o,\xi,\Box\Diamond S]^{\sigma_f}}[X_{(r',\gamma)}] \\
&\geq \mathbb{E}_{M^{\sigma_f}}[X_{(r,\gamma)}] + \tau \\
&\geq (v_\epsilon + \epsilon/2) + \tau \\
&\geq (v_\epsilon + \epsilon/2) - (\epsilon/2) = v_\epsilon.
\end{aligned}$$

This completes the proof of the theorem. □

C Details on experiments

C.1 Dubin's RTA scenarios

Model All of the aircarfts follow the dynamical model given by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos(\theta)\cos(\psi) \\ v\sin(\theta)\cos(\psi) \\ v\sin(\psi) \\ \omega \\ \Psi \\ a \end{bmatrix}$$

where for the state: (x, y, z) is the position, θ is the heading angle, ψ is the pitch angle, and v is the velocity; and for the input: ω is the change in heading rate, Ψ is the change in pitch rate, and a is the acceleration. The lead aircraft follows a set circular path where $a = 0$, $\Psi = 0$, and ω is constant. The follower aircraft attempt to follow at some position relative to the leader using a tracking controller given by

$$\begin{bmatrix} \omega \\ \Psi \\ a \end{bmatrix} = \begin{bmatrix} \omega_{\text{ref}} \\ K_1(\psi_{\text{ref}} - \psi) \\ K_2(v_{\text{ref}} - v) \end{bmatrix} \tag{1}$$

where $K_1, K_2 > 0$ and ω_{ref}, ψ_{ref}, and v_{ref} are the reference change in heading rate, reference pitch angle, and reference velocity respectively. The untrusted controller U follows the tracking controller in (1) exactly, and the safety controller S follows the tracking controller but sets $a = 0$. The positions U tracks is closer to the leader, and the position that S tracks is further from the leader.

Dubins In the Dubins example, the lead aircraft follows a circular path with an angular velocity between 0.6 and 0.7 rad/s, and a velocity between 450 and 550. The reach region is 400 units behind the lead and 100 units to its left. The follower starts 250 units behind the leader, and has the same velocity as the leader.

Dubins+O In the Dubins+Oexample, the lead aircraft follows a circular path with an angular velocity of 0.2 rad/s and a velocity of 500. The follower starts 500 units to the left of the leader, and has an initial velocity of 400, to match the angular velocity of the leader. The obstacles are cubes with side length 100. The first cube is offset 250 units radially inward from the leader's path so that it's center coincides with the tracking point of the follower's untrusted controller. It appears at an angle along the path between 2 and 4.71 radians. The second obstacle has a radial offset of 500 to correspond with the follower's safety controller tracking point, and appears at an angle along the path of 0.78 radians.

Fleet In the Fleet example, the leader follows a circular path with angular velocity between 0.3 and 0.4 rad/s with a velocity between 450 and 650. The followers begin on the tracking point of their safety controllers, so the first follower starts 250 units behind and 200 units to the left of the leader, while the second follower starts 250 units behind and 200 units to the right of the leader. They share a untrusted controller tracking point 250 units behind the leader. The reach region they must enter infinitely often is centered on this point with a radius of 100.

C.2 Optimal orbit transfer

Model The satellite follows the 2-dimensional Clohessy Whiltshire dynamics given by

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \ddot{x}\\ \ddot{y}\end{bmatrix} = \begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ 3n^2 & 0 & 0 & 2n\\ 0 & 0 & -2n & 0\end{bmatrix}\begin{bmatrix}x\\ y\\ \dot{x}\\ \dot{y}\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & 0\\ 1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}u_x\\ u_y\end{bmatrix}$$

where n is the orbital rate. A zero-input trajectory of the satellite is called a natural motion trajectory (NMT). The explicit solution to a NMT is given by

$$x(t) = 4x_0 + 2\frac{\dot{y}_0}{n} - (3x_0 + 2\frac{\dot{y}_0}{n})\cos(n(t-t_0)) + \frac{\dot{x}_0}{n}\sin(n(t-t_0))$$
$$y(t) = y_0 - 2\frac{\dot{x}_0}{n} + 2\frac{\dot{x}_0}{n}\cos(n(t-t_0)) + (6x_0 + 4\frac{\dot{y}_0}{n})\sin(n(t-t_0)) + (3nx_0 + 2\dot{y}_0)(t-t_0)$$
$$\dot{x}(t) = \dot{x}_0\cos(n(t-t_0)) + (3nx_0 + 2\dot{y}_0)\sin(n(t-t_0))$$
$$\dot{y}(t) = (6nx_0 + 4\dot{y}_0)\cos(n(t-t_0)) - 2\dot{x}_0\sin(n(t-t_0)) - (6nx_0 + 3\dot{y}_0)$$

where x_0, y_0, $\dot{x}_0$, and $\dot{y}_0$ are the initial conditions. The explicit solutions can be rewritten in terms of the NMT parameters a, d, c, and α where

$$a = [(3x_0 + 2\frac{\dot{y}_0}{n})^2 + (\frac{\dot{x}_0}{n})^2]^{\frac{1}{2}}$$
$$d = y_0 - 2\frac{\dot{y}_0}{n}$$
$$c = 2x_0 + \frac{\dot{y}_0}{n}$$
$$\cos(\alpha) = -\frac{1}{a}(3x_0 + 2\frac{\dot{y}_0}{n}) \text{ and } \sin(\alpha) = -\frac{\dot{x}_0}{na}$$

such that

$$x(t) = 2c + a\cos(n(t-t_0) + \alpha)$$
$$y(t) = d - 3nc(t-t_0) - 2a\sin(n(t-t_0) + \alpha)$$
$$\dot{x}(t) = -an\sin(n(t-t_0) + \alpha)$$
$$\dot{y}(t) = -3nc - 2an\cos(n(t-t_0) + \alpha).$$

The NMT is periodic with period $T = \frac{2\pi}{n}$ with ellipse with eccentricity $\frac{\sqrt{3}}{2}$ centered at $(0, d)$ when $c = 0$. The parameter a represents the size of the ellipse and α indicates the initial position on the ellipse.

Orbit transfer scenario In the orbit transfer scenarios, spacecraft begin in the NMT (10, 5, 0) and must transfer to the orbit (50, 10, 0). In scenarios with an obstacle, the obstacle's NMT is (25, 0, 0). A collision occurs if the spacecraft is within 5 units of the obstacle.

D Reinforcement Learning Hyperparameters

Hyperparameters	
Shared	
γ	0.995
λ	0.995
Layers (RTA)	2
Layers (Satellite)	4
Hidden units (RTA)	512
Hidden units (Satellite)	1024
Learning rate	5.00E-05
Optimizer	Adam
β_1	0.9
β_2	0.999
Max Episode Length (RTA)	400
Max Episode Length (Satellite)	80
Activation function	Tanh
PPO	
Batch size	200000
Batch mode	complete episodes
Max SGD iterations	20
ϵ	0.3
Value function clipping	10
SAC	
Entropy learning rate	1.00E-05
Critic learning rate	0.005
Actor learning rate	0.005
Target update frequency	1024
Steps before learning	1500
τ	1
Batch mode	truncate episodes
CPO	
Cost limit	1
Batch size	40000
Max SGD iterations	80