Robotics: Science and Systems 2021
Held Virtually, July 12–16, 2021

Entropy-Guided Control Improvisation

Marcell Vazquez-Chanlatte*, Sebastian Junges*, Daniel J. Fremont†, Sanjit A. Seshia*
University of California, {Berkeley*, Santa Cruz†}

***Abstract*—High level declarative constraints provide a powerful (and popular) way to define and construct control policies; however, most synthesis algorithms do not support specifying the degree of randomness (unpredictability) of the resulting controller. In many contexts, e.g., patrolling, testing, behavior prediction, and planning on idealized models, predictable or biased controllers are undesirable. To address these concerns, we introduce the *Entropic Reactive Control Improvisation* (ERCI) framework and algorithm which supports synthesizing control policies for stochastic games that are declaratively specified by (i) a *hard constraint* specifying what must occur, (ii) a *soft constraint* specifying what typically occurs, and (iii) a *randomization constraint* specifying the unpredictability and variety of the controller, as quantified using causal entropy. This framework, which extends the state of the art by supporting arbitrary combinations of adversarial and probabilistic uncertainty in the environment. ERCI enables a flexible modeling formalism which we argue, theoretically and empirically, remains tractable.**

I. Introduction

The use of declarative specifications, e.g. in the form of temporal logic formulas, has become a popular way to construct high-level robot controllers [30, 58, 27, 21, 28, 40, 33]. Given a user provided specification, *synthesis* algorithms aim to automatically create a control policy that ensures that the specification is met, or explain why such a policy does not exist. Together, synthesis and declarative specifications facilitate quickly and intuitively solving a wide variety of control tasks. For example, consider a delivery drone operating in a workspace. One may specify the drone should "within 10 minutes, visit four locations (in any order) *and* avoid crashing.". A synthesis tool may then create a finite state controller which guarantees this specification is met, under a particular world model. Importantly, while many controllers may conform to the provided specification, many synthesis algorithms provide a single, often deterministic, policy. For instance, in our drone example, a synthesized controller may generate only a single path through the workspace.

In some settings, such policies are undesirable. First, in many tasks, the predictability (or bias) of the policy may be a liability. Examples include patrolling [3], behavior prediction and inference [57], and creating controller harnesses for fuzz testing (see motivating example in Sec. II). Second, synthesis algorithms work on *idealized* models, and thus any policy that overcommits to any given model quirk may in practice yield poor performance. In such settings, randomization is known to make policies more robust against worst-case deviations [60, 17]. Unfortunately, classical synthesis methods result in policies that need not (and typically do not) exhibit randomization.

To address these potential deficits, we advocate for the adoption of the recently proposed *control improvisation* [19, 20] framework, in which one specifies a controller with three types of declarative constraints. (i) *Hard constraints* that, as in the classical setting, must hold on every execution, (ii) *soft constraints* that should hold on most executions, and (iii) *randomization constraints* that ensure that a synthesized policy does not overcommit to a particular action or behavior. The key challenge when solving control improvisation is that randomization and performance, in the form of soft constraints, constitute a natural trade-off.

So far, control improvisation has only be studied in non-deterministic domains where uncertainty is resolved adversarially [19]. This assumption is often too restrictive and leads (together with the soft/hard constraints) to conservative policies or common situations in which the synthesis algorithm cannot be employed at all. To overcome this weakness, we develop a theory of control improvisation in stochastic games which admit *arbitrary combinations of nondeterministic and probabilistic uncertainty*, including unknown or imprecise transition probabilities.

Technically, we formulate our problem on *simple stochastic games* [14], an extension of *Markov decision processes* (MDPs) that divides states between controllable states and uncontrollable (or adversarially controlled) states. *Soft constraints* are finite horizon temporal properties with a threshold on the worst-case probability of the property holding by the end of the episode. *Hard constraints* are soft constraints to be satisfied with probability 1. In contrast to other work on control improvisation, we adopt causal entropy as a natural means to formalize *randomness constraints*. Causal entropy is a prominent notion in directed information theory [39] that strongly correlates with robustness in the (inverse) reinforcement learning setting [60, 17]. We refer to this variant of control improvisation as *Entropic Reactive Control Improvisation* (ERCI) and show that ERCI conservatively extends reactive control improvisation [19] to stochastic games. More precisely, entropy can be used in the non-stochastic setting and yields results analogous to reactive control improvisation. ERCI also extends classical policy synthesis in stochastic games, i.e. synthesis in absence of randomness constraints as, e.g., implemented in PRISM-games [35].

Contributions. In summary, this paper contributes ERCI, an algorithmic way to trade performance and randomization in stochastic games. As we motivate in the example below, games that combine both adversarial and probabilistic behavior in an environment allow for modeling flexibility, facilitating applicability to new domains. To support this extension, the paper proposes and shows the benefits of formulating

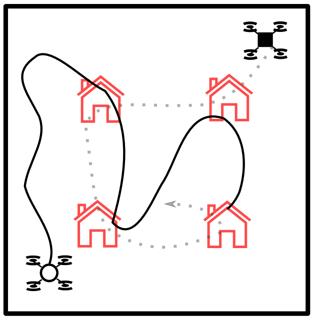


Fig. 1. Illustration of delivery drone testing example. The goal is to synthesize a policy for the bottom left (white circle) drone to test the controller of the top right (black square) drone. Ideally, the synthesized policy should be as randomized as possible to avoid testing bias.

randomization constraints with causal entropy. Finally, this work contributes the necessary technical machinery and a prototype implementation. Combined, our theoretical and empirical analysis suggest that the ERCI framework contributes a tractable and flexible modeling formalism.

Overview. This paper is structured as follows. We begin with a motivating example (Sec. II). Then we provide preliminaries and formalize the ERCI problem statement (Sec. III). Next, we cast ERCI as a multi-objective optimization problem and study properties of the solution set (Sec. IV). With this technical machinery developed, Sec. V re-frames existing literature on maximum causal entropy inference and control to derive an algorithm for MDPs. Then in Sec. VI, we provide an algorithm for the general case of stochastic games. We conclude with an empirical evaluation (Sec. VII) and a comparison with related work, e.g., other control improvisation formulations (Sec. VIII). Proofs are attached in Sec. X.

II. Motivating Example

We consider a scenario in which a regulatory agency wishes to certify the safety and performance of a new delivery drone D_{new}. As part of the process, the agency runs D_{new} through a series of tests. For example, given a certain delivery route, the agency investigates whether D_{new} successfully delivers packages while avoiding *other* delivery drones. To execute this test, the agency decides to synthesize a controller for another delivery drone, D_{test}, to test if D_{new} can be certified.

Concretely, suppose we command D_{new} to continuously visit four houses in some workspace. We illustrate such a scenario in Fig. 1, in which D_{new} and D_{test} are shown as black square and white circle drones respectively. For this test scenario, the regulatory agency, wishes to exam how D_{new} responds to delivering packages to the red houses in the presence of D_{test}. In particular, it would like to let D_{test} also deliver packages while avoiding D_{new}. Importantly, to properly exercise D_{new}, D_{test} should show a *variety* of behaviors meeting the specification, and the behaviors should not be biased to any behavior beyond the given specification.

With the ERCI framework, the agency may formalize the above scenario with the following constraints on D_{test}:

1) (*hard constraint*) Ensure that the two drones *never* collide.
2) (*soft constraint*) With probability at least .8, visit all four houses within 10 minutes.
3) (*randomness constraint*) Perform this task as unpredictably as possible.

What remains is to synthesize a controller given the constraints *and* the world model. At this point, it is worth examining more closely how one models D_{new}'s controller when synthesizing D_{test}. We illustrate by examining three models. In all models, we capture the behaviors of D_{new} and D_{test}. We focus on D_{new}, but the ideas carry over to modeling the actuation of D_{test}.

Nondeterministic Model. The simplest approach to modeling is not to make any assumptions about D_{new} beyond what already has been established. Here, we model that the houses are visited either in clockwise or counter-clockwise order but that it may switch direction at *any time*. Such a model is too liberal and our assumptions under which we plan the behavior for D_{test} is too pessimistic, which leads to a bad test set. First, if D_{new} is unrestricted, then D_{test}'s behavior is severely limited, as it must behave conservatively to avoid collisions under all possible motions by D_{new} (even very unlikely motions). This limitation restricts the variance of its behavior, and it will not test D_{new}'s true behavior. A purely non-deterministic model for D_{new} thus may not lead to the synthesis of adequate behavior for D_{test}.

Stochastic Model. Rather than the pessimistic nondeterministic (or adversarial) assumption, we may collect data about D_{new} and construct a stochastic model, e.g., using inverse reinforcement learning [43]. Concretely (but simplified), after examining the data, one observes that D_{new} appears to flip a biased coin with fixed probability p whenever it reaches a house to decide whether or not to turn around. This models D_{new} much more precisely, and allows for more targeted test by D_{test}.

Nondeterministic and Stochastic Model. However, a natural criticism for stochastic models is the dependence on *fixed* probabilities. Obtaining such probabilities with confidence requires many tests which defeat the purpose of our test setup, and making point-estimates from little data may not create faithful models of the actual behavior. In absence of enough (or reliable) data, we can arbitrarily combine nondeterministic choices and stochastic behavior. We may use stochastic abstractions for parts that we can faithfully model, and nondeterministic behavior in absence of data. In particular, we support interval-valued transition probabilities. Consider the delivery-drone D_{new}. Rather than inferring a point-estimate from data, we may have inferred that the probability of turning around is in the interval $[p-\varepsilon, p+\varepsilon]$ for adequate values of p and ε. Furthermore the actual probability may even depend on aspects of the current state.

ERCI as a unifying framework. The strength of the (entropy-guided) control improvisation framework is that we can combine all these aspects into a single and thus flexible computational model. In particular, the models above are captured by a 2-player game, a 1.5-player game (MDP) and a 2.5-player game (stochastic game, SG), respectively. In all

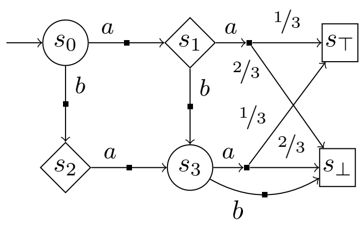


Fig. 2. A running example.

cases, the first player controls the behavior of D_{test} and this controller is to be synthesized. We contribute an algorithm that synthesizes a controller that maximally randomizes in all of the formalisms discussed above. In the coming sections, we shall formally define the ERCI problem, highlight that there is an implicit trade-off between performance of the soft constraint and unpredictability, and provide an algorithm solving ERCI for SGs.

III. Problem Statement

This section formalizes the novel Entropic Reactive Control Improvisation (ERCI) problem. We start with some necessary definitions and notations on stochastic games.

A. Stochastic Games

A (2.5-player) *stochastic game* (SG) is a tuple $\mathcal{G} = \langle S, \iota, A, P \rangle$. The finite set of *states* $S = S_{\mathsf{ego}} \cup S_{\mathsf{env}}$ is partitioned into a set S_{ego} of (controlled) ego-states and a set S_{env} of (uncontrolled) env-states. $\iota \in S_{\mathsf{ego}}$ is the *initial state*, A is a finite set of *actions*, and $P \colon S \times A \to \mathit{Distr}(S)$ is the *transition function*. For simplicity of exposition, we assume w.l.o.g. that controlled and uncontrolled states alternate. Thus, P is defined by two *partial* transition functions: $P_{\mathsf{ego}} \colon S_{\mathsf{ego}} \times A \to \mathit{Distr}(S_{\mathsf{env}})$, $P_{\mathsf{env}} \colon S_{\mathsf{env}} \times A \to \mathit{Distr}(S_{\mathsf{ego}})$. We identify the available actions[1] as $A(s) \stackrel{\text{def}}{=} \{\alpha \mid P(s,\alpha) \neq \bot\}$. States without available actions, i.e., states with $A(s) = \emptyset$ are called *terminal states*. The *successor states* of a state s and an (enabled) action α is the set of states that are reached from s within one step with a positive transition probability, i.e., $\mathsf{Succ}(s,\alpha) \stackrel{\text{def}}{=} \{s' \mid P(s,\alpha)(s') > 0\}$, and $\mathsf{Succ}(s) \stackrel{\text{def}}{=} \bigcup_{\alpha \in A(s)} \mathsf{Succ}(s,\alpha)$.

Example 1. We introduce a six-state toy-example (Fig. 2) to illustrate the definitions. Terminal states are drawn with a rectangle, ego-states with a circle and env-states with a diamond. For every state s and action α, we draw transitions in the form of edges that connect all successors s', and label them with the associated probabilities $P(s,\alpha)(s')$. For conciseness, we omit labelling probability 1 transitions.

SGs capture a variety of models. For example, if $|A(s)| = 1$ for all uncontrolled states, $s \in S_{\mathsf{env}}$, then $\mathcal{G}$ is a *Markov decision process* (MDP). If $|A(s)| = 1$ for all $s \in S$, then $\mathcal{G}$ is a *Markov chain*. If $P(s,\alpha)$ is a Dirac distribution for every $s \in S$ and $\alpha \in A$, then $\mathcal{G}$ is called *deterministic* or a *2-player game*.

[1]We use a partial function as we explicitly allow modeling unavailable actions, e.g., we can model that a door can only be opened when close enough to the door.

B. Paths and Path Properties

A finite *path*, ξ, of length n is a sequence $s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} s_2 \to \ldots \to s_n$ in $(S \times A)^n \times S$ where $P(s_i,\alpha_i)(s_{i+1}) > 0$ for each i. We denote the length with $|\xi|$, and denote s_n, i.e., the last element of ξ, with $\mathsf{last}(\xi)$. Further, note that ego states are even indexed and env states are odd indexed as we assume alternation. A path, $\xi' = s'_0 \xrightarrow{\alpha'_0} \ldots$, is a *prefix* of ξ, if for all $i \le |\xi'|$, $s_i = s'_i$ and for all $i < |\xi'|$, $\alpha_i = \alpha'_i$. The set of all finite paths of length n is denoted $\mathsf{Paths}^{\mathcal{G}}_n$, and $\mathsf{Paths}^{\mathcal{G}} = \bigcup_{n \in \mathbb{N}} \mathsf{Paths}^{\mathcal{G}}_n$. We omit $\mathcal{G}$ whenever it is clear from the context. It is helpful to partition paths based on their last state: $[\mathsf{Paths}]_{\mathsf{ego}} = \{\xi \in \mathsf{Paths} \mid \mathsf{last}(\xi) \in S_{\mathsf{ego}}\}$ and $[\mathsf{Paths}]_{\mathsf{env}} = \mathsf{Paths} \setminus [\mathsf{Paths}]_{\mathsf{ego}}$.

Example 2. In Fig. 2, there are two paths that end in s_3, $s_0 \xrightarrow{a} s_1 \xrightarrow{b} s_3$ and $s_0 \xrightarrow{b} s_2 \xrightarrow{a} s_3$, both of length 2. Both paths are in $[\mathsf{Paths}]_{\mathsf{ego}}$, as $s_3 \in S_{\mathsf{ego}}$.

Whenever some state s is reached, the corresponding player draws an action from $A(s)$. As standard, we capture this with the notion of a scheduler[2]. A *scheduler* is a tuple of *player policies* $\sigma = \langle \sigma_{\mathsf{ego}}, \sigma_{\mathsf{env}} \rangle$ with $\sigma_{\mathsf{i}} \colon [\mathsf{Paths}]_{\mathsf{i}} \to \mathit{Distr}(A)$ such that $\mathsf{support}(\sigma_i(\xi)) \subseteq A(\mathsf{last}(\xi))$ for each ξ, i.e., for every history, the policy sets a distribution over the enabled successor actions. For a given path, ξ and a policy σ_{i}, we denote by $\sigma_{\mathsf{i}}(\alpha \mid \xi)$ the distribution of actions induced by σ_{i} given the path ξ. To ease notation, we liberally use the notation $\sigma \colon \mathsf{Paths} \to \mathit{Distr}(A)$, where this function is given dependent on which player owns the last state.

Example 3. An example for a ego-policy σ_{ego} is given by,

$$\sigma_{\mathsf{ego}}(\alpha \mid \xi) = \begin{cases} 1/2 & \text{if } \alpha \in \{a,b\},\ \xi = s_0, \\ 1 & \text{if } \alpha = a,\ \xi = s_0 \xrightarrow{b} s_2 \xrightarrow{a} s_3, \\ 1 & \text{if } \alpha = b,\ \xi = s_0 \xrightarrow{a} s_1 \xrightarrow{b} s_3. \end{cases}$$

The probability $\Pr(\xi \mid \sigma)$ of a finite path ξ in an SG $\mathcal{G}$ conditioned on a policy σ is given by the product of the transition probabilities along a path. More precisely, we define the probability $\Pr(\xi \mid \sigma)$ recursively as:

$$\begin{aligned} &\Pr(s \mid \sigma) \stackrel{\text{def}}{=} 1 \\ &\Pr(\xi \mid \sigma) \stackrel{\text{def}}{=} \Pr(\xi' \mid \sigma) \cdot \sigma(\alpha \mid \xi') \cdot P(\mathsf{last}(\xi'),\alpha)(s') \end{aligned} \tag{1}$$

where $\xi = \xi' \xrightarrow{\alpha} s'$. The probability of a prefix-free set $X \subseteq \mathsf{Paths}$ of paths is the sum over the individual path probabilities, $\Pr(X \mid \sigma) = \sum_{\xi \in X} \Pr(\xi \mid \sigma)$.

Next, we develop machinery to distinguish between desirable and undesirable paths. We focus on finite path properties, referred to as specifications or constraints, that are decidable within some fixed $\tau \in \mathbb{N}$ time steps, e.g., "Recharge before t=20." Technically, we represent these path properties as prefix free sets of finite paths, φ, reflecting some formal property[3].

[2]Also known as *strategy* or *policy*.

An example are all paths that end in a particular terminal state $s_\top$ within τ steps.

C. Control Improvisation

In control improvisation, we aim to find an ego-policy, σ_{ego}, that satisfies a combination of hard- and soft constraints, and additionally generates surprising behavior, where we measure the expected surprise by the causal entropy [39] over the paths.

We first define causal entropy on arbitrary sequences of random variables. Let $\mathcal{X}_{1:i} \stackrel{\text{def}}{=} \mathcal{X}_1, \dots, \mathcal{X}_i$ and $\mathcal{Y}_{1:i} \stackrel{\text{def}}{=} \mathcal{Y}_1, \dots, \mathcal{Y}_i$ denote two sequences of random variables. The probability of $\mathcal{X}_{1:i}$ causally conditioned on $\mathcal{Y}_{1:i}$ is:

$$\Pr(\mathcal{X}_{1:i} \mid\mid \mathcal{Y}_{1:i}) \stackrel{\text{def}}{=} \prod_{j=1}^{i} \Pr(\mathcal{X}_j \mid \mathcal{X}_{1:j-1}\mathcal{Y}_{1:j}). \tag{2}$$

The causal entropy of $\mathcal{X}_{1:i}$ given $\mathcal{Y}_{1:i}$ is then defined as,

$$H(\mathcal{X}_{1:i} \mid\mid \mathcal{Y}_{1:i}) \stackrel{\text{def}}{=} \mathbb{E}_{\mathcal{X}_{1:i},\mathcal{Y}_{1:i}}[-\log(\Pr(\mathcal{X}_{1:i} \mid\mid \mathcal{Y}_{1:i}))] \tag{3}$$

Using the chain rule, one can relate causal entropy to (non-causal) entropy, $H(\mathcal{X}|\mathcal{Y}) \stackrel{\text{def}}{=} \mathbb{E}_{\mathcal{X}}[-\log(\Pr(\mathcal{X} \mid \mathcal{Y}))]$ via:

$$H(\mathcal{X}_{1:i} \mid\mid \mathcal{Y}_{1:i}) = \sum_{t=1}^{i} H(\mathcal{X}_t \mid \mathcal{Y}_{1:t}, X_{1:t-1}) \tag{4}$$

This relation shows that: (1) Causal entropy is always lower bounded by non-causal entropy (and thus non-negative). (2) Causal entropy can be computed "backward in time". (3) Causal and non-causal conditioning can be mixed,

$$H(\mathcal{X}_{1:i} \mid\mid \mathcal{Y}_{1:i} \mid Z) \stackrel{\text{def}}{=} \sum_{t=1}^{i} H(\mathcal{X}_t \mid \mathcal{Y}_{1:t}, X_{1:t-1}, Z). \tag{5}$$

Intuitively, and contrary to non-causal entropy, causal entropy does *not* condition on variables that have not been revealed, e.g., on events in the future. This makes causal entropy particularly well suited for measuring predictability in *sequential* decision making problems, as the agents cannot observe the future [60].

We now define causal entropy in stochastic games. Recall that a path alternates states and actions. The next state after observing a sequence of state-action pairs is a random variable. Formally, given $\mathcal{G}$ and a scheduler σ, let us denote by $\mathcal{A}^{\mathsf{ego}}_{1:i}$ and $\mathcal{S}_{1:i}$ random variable sequences for ego-player actions and states respectively. The causal entropy of controllable actions in τ-length paths under σ is then,

$$H_\tau(\sigma) \stackrel{\text{def}}{=} H(\mathcal{A}^{\mathsf{ego}}_{1:\tau'} \mid\mid \mathcal{S}_{1:\tau}), \tag{6}$$

where $\tau' = \lceil \frac{\tau}{2} \rceil$ is the number of ego-actions due to alternation.

Example 4. Consider the uniform ego policy on Fig. 2. If $\sigma_{\mathsf{env}}(a \mid \xi) = 1$. $H_\tau(\sigma) = \log(2) + {}^1\!/_2(\log(2))$. Note, only ego can *add* entropy, while env and stochastic transitions yield convex combinations via expectation.

We now formalize the problem statement.

[3]Such paths may e.g. be defined using temporal properties such as linear temporal logic over finite traces (LTLf) [24].

The Entropic Control Improvisation (ERCI) Problem: Given a SG $\mathcal{G}$, τ-bounded path properties ψ and φ, and thresholds $\mathbf{p} \in [0,1]$ and $\mathbf{h} \in [0,\infty)$, find a ego-policy σ_{ego} (or report that none exists) such that for every env-policy σ_{env},

1) (*hard constraint*) $\Pr(\psi \mid \sigma) \geq 1$
2) (*soft constraint*) $\Pr(\varphi \mid \sigma) \geq \mathbf{p}$
3) (*randomness constraint*) $H_\tau(\sigma) \geq \mathbf{h}$

where $\sigma = \langle \sigma_{\mathsf{ego}}, \sigma_{\mathsf{env}} \rangle$.

We say that an instance of the ERCI problem is realizable, if an appropriate σ_{ego} exists and call such σ_{ego} an *improviser*. The problem is unrealizable otherwise.

IV. ERCI as multi-objective optimization

We investigate the ERCI problem statement. Based on a sequence of observations, we reduce the ERCI problem to the Core ERCI problem which significantly eases the description (and implementation) of the algorithm afterwards.

A. Preprocessing

To ease the technical exposition, without loss of generality, we make the following assumptions: We assume the graph structure underlying the SG is finite and acyclic – and thus all paths are finite length. When considering τ-bounded path properties (monitorable by finite automata), this assumption is naturally realized by a τ-step unrolling of a monitor augmented SG [4], i.e., augmenting the state space with a counter from 0 to τ and the current property monitor state.

Next, in order to ensure the hard constraint, ψ, we calculate all states from which the env-player can enforce violating the hard constraint. Such states are identifiable using a single topologically ordered pass over $\mathcal{G}$ from the terminal states to the initial state. We remove such states along with their in- and outgoing transitions. Any ego-policy now satisfies the hard constraint. The remaining terminal states are all merged into two states $s_\top$ and $s_\bot$, based on membership in φ, i.e.,

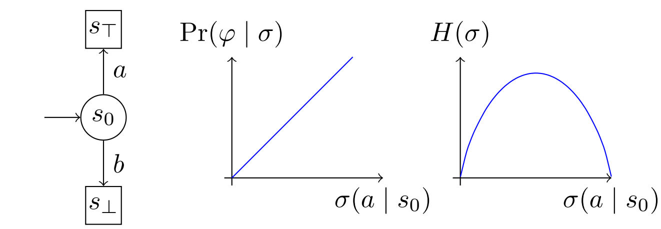
$$\begin{aligned} \mathsf{last}(\xi) = s_\top &\implies \xi \in \varphi \\ \mathsf{last}(\xi) = s_\bot &\implies \xi \notin \varphi \end{aligned}. \tag{7}$$

Example 5. In Fig. 3a we show a (deterministic) MDP and we plot for all schedulers the induced probability to reach $s_\top$ and the induced causal entropy, in Fig. 3b and 3c, respectively. We see that taking action a with increasing probability yields a larger probability to reach $s_\top$, whereas taking action a and b uniformly at random is optimal for the entropy.

B. Geometric Perspective

There is a natural trade-off between probability of generating paths in φ (from here onwards: *the performance*) and causal entropy induced by a policy (*the randomization*). In particular, with all other ingredients fixed, we are interested in understanding the combinations of $\mathbf{p}$ and $\mathbf{h}$ that yield a

[4]One may then represent this unrolled graph as a binary decision diagram, resulting in a (typically) concise graph that grows proportional to the horizon and minimal state space augmentation required [57].



(a) Minimal MDP (b) Probability to reach $s_\top$ (c) Causal Entropy

Fig. 3. Minimal ERCI problem with $\varphi = (\mathsf{last}(\xi) = s_\top)$

solvable instance of the (core) ERCI problem. To this end, we cast ERCI as an instance of a multi-objective optimization problem, and study its Pareto front. Some ideas are inspired by variants of multi-objective analysis of MDPs with multiple soft constraints, e.g. [11, 16, 18].

It is convenient to consider this front geometrically. To begin, given a fixed ERCI instance, a scheduler σ *induces* a point x_σ:

$$x_\sigma \stackrel{\text{def}}{=} \Big\langle \Pr(X_\varphi \mid \sigma), H(\sigma) \Big\rangle \in [0,1] \times [0,\infty). \tag{8}$$

To ease notation, for $x_\sigma = \langle p, h \rangle$ we use $p_\sigma \stackrel{\text{def}}{=} p$ and $h_\sigma \stackrel{\text{def}}{=} h$. Next, we partially order these points via the standard product ordering:

$$\langle p, h \rangle \preceq \langle p', h' \rangle \quad \text{ iff } \quad p \le p' \wedge h \le h'. \tag{9}$$

We say that σ_{ego} *guarantees* a point $x_{\mathsf{ego}} \stackrel{\text{def}}{=} \langle p, h \rangle$, if for every policy σ_{env}, using $\sigma = \langle \sigma_{\mathsf{ego}}, \sigma_{\mathsf{env}} \rangle$, we have $p_\sigma \ge p$ and $h_\sigma \ge h$. Thus, a point is guaranteed if no matter what policy env uses, x_σ will induce a point no worse w.r.t. to either randomization or performance than x_{ego}. We define *the set of guaranteed points* for a scheduler σ_{ego}:

$$\mathbb{S}[\sigma_{\mathsf{ego}}] \stackrel{\text{def}}{=} \{ \langle p, h \rangle \mid \sigma_{\mathsf{ego}} \text{ guarantees } \langle p, h \rangle \}. \tag{10}$$

We observe that guaranteed points are downward closed, i.e., if σ_{ego} guarantees x and $x' \preceq x$, then σ_{ego} guarantees x'.

Example 6. Consider Fig. 4a. We fix σ_{ego} and in the blue hatched area draw all points induced by $\sigma = \langle \sigma_{\mathsf{ego}}, \sigma_{\mathsf{env}} \rangle$ when varying σ_{env}. We take the minimal randomness h and the minimal performance p. The points in the downward closure of $\langle p, h \rangle$ (green circle) are the guaranteed points for σ_{ego} in the green solid area. We notice the gap between both areas: While the performance and randomization may be better than the optimum that ego can guarantee, it cannot guarantee a higher randomization *and* performance simultaneously, as the env-player would have a counter-policy violating either the performance *or* the randomization.

Points guaranteed by some σ_{ego} are called *achievable*. Thus, the achievable points are: $\mathbb{S} = \bigcup_{\sigma_{\mathsf{ego}}} \mathbb{S}[\sigma_{\mathsf{ego}}]$. Importantly, the ERCI problem is realizable iff $\langle \mathbf{p}, \mathbf{h} \rangle$ is achievable. Thus, to solve ERCI instances, we start by characterizing $\mathbb{S}$. We start by observing the $\mathbb{S}$ is convex[5] (proof in Sec X).

Proposition 1. *The set of achievable points, $\mathbb{S}$, is convex.*

[5]That is, $x, x' \in \mathbb{S}$ implies for every $w \in [0,1]$ that $w \cdot x + (1-w) \cdot x \in \mathbb{S}$

Next, because $\mathbb{S}$ is downward closed, it suffices to study the "maximal" or non-dominated points. Precisely, we say that a point x is *dominated* by x' if $x \prec x'$, i.e., if $x \preceq x' \wedge x \neq x'$. The Pareto front $\mathcal{F}_{\mathbb{S}}$ of $\mathbb{S}$ is then the set of non-dominated achievable points,

$$\mathcal{F}_{\mathbb{S}} \stackrel{\text{def}}{=} \{ x \in \mathbb{S} \mid \forall x' \in \mathbb{S}, x \not\prec x' \}. \tag{11}$$

Importantly, it holds that the ERCI problem is satisfiable iff there exists a $x \in \mathcal{F}_{\mathbb{S}}$ such that $\langle \mathbf{p}, \mathbf{h} \rangle \preceq x$.

Example 7. The set $\mathbb{S}$ illustrated in Fig. 4b is obtained by taking the union of guaranteed points, and can be characterized by the set of points on the Pareto front: This is the curved border between the green and white area, in particular the three green dots are on the Pareto front. Any ERCI instance with $\langle \mathbf{p}, \mathbf{h} \rangle$ in the green area is realizable.

Approximating the Pareto front gives a natural approximation scheme for ERCI instances: For any subset $\mathcal{F} \subseteq \mathcal{F}_{\mathbb{S}}$,

1) If there exists an $x \in \mathcal{F}$ such that $\langle \mathbf{p}, \mathbf{h} \rangle \preceq x$, then the ERCI problem must be realizable and x is a witness to realizability.
2) If there exists an $x \in \mathcal{F}$ such that $x \prec \langle \mathbf{p}, \mathbf{h} \rangle$, then the ERCI problem is not realizable and x is a witness to unrealizability.

Due to convexity, we may speed up the search for realizability: If there exist $x_1, x_2 \in \mathcal{F}$ such that $\langle \mathbf{p}, \mathbf{h} \rangle \prec \big(w \cdot x_1 + (1-w) \cdot x_2 \big)$, we call x_1, x_2 a witness-pair.

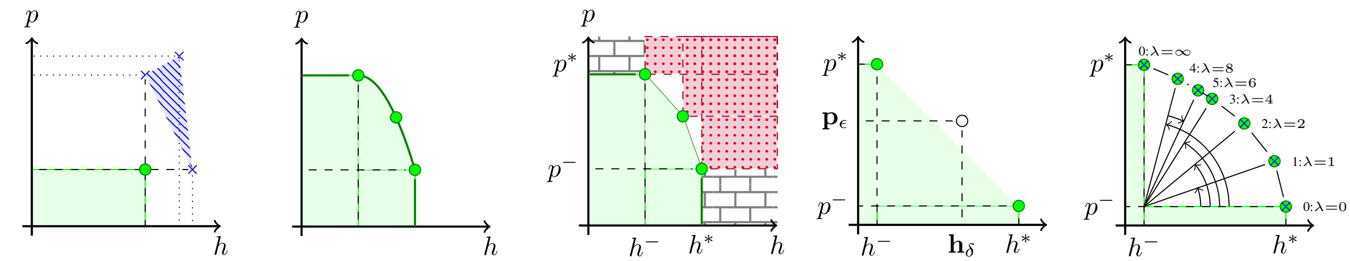
Remark 1. Given a witness(pair) to realizability, it is easy to extract the corresponding improviser. Let x_1, x_2 be a witness-pair to realizability, induced by σ_{λ_1} and σ_{λ_2} such that $\langle \mathbf{p}, \mathbf{h} \rangle \preceq w \cdot x_1 + (1-w) \cdot x_2$, then the policy described by

$$\sigma^*_{\mathsf{ego}}(\alpha \mid s) \stackrel{\text{def}}{=} w \cdot \sigma_{\lambda_1}(\alpha \mid s) + (1-w) \cdot \sigma_{\lambda_2}(\alpha \mid s) \tag{12}$$

is an improviser solving the ERCI problem.

Example 8. Consider Fig. 4c. We have found three points on the Pareto front, and already have a good impression of the trade-off between randomization and performance. In particular, the green area is definitively a subset of $\mathbb{S}$: It exploits the downward closure and the convexity of $\mathbb{S}$. The red (dotted) part contain the points on the Pareto front in their downward closure, thus they cannot be part of the Pareto front themselves. Furthermore, the topmost point on the Pareto front was obtained by maximizing performance (and optimizing randomization only as a secondary objective). Thus, by construction, the bricked area at the top is not realizable. Analogously, the bricked area at the right reflects non-achievable randomization.

Remark 2. We notice that the multi-objective optimization perspective allows us to extend the set of witnesses for unrealizability. In particular, every point of the Pareto-curve can be described as optimizing some scalarization of the objectives. Geometrically, it optimizes along a particular direction. Whenever we know that a Pareto-optimal point $x = \langle p, h \rangle$ optimizes a weighted objective with weights $w = \langle w_1, w_2 \rangle$,



(a) Guaranteed points $\mathbb{S}_{\sigma_{\text{ego}}}$ (b) Solutions $\mathbb{S}$ (c) Iterative construction (d) Regret-based ERCI (e) Rationality-based algorithm

Fig. 4. Geometric interpretation of the ERCI problem for some fixed SG.

then x and w *together* are a witness for unrealizability for $\langle \mathbf{p}, \mathbf{h} \rangle$ whenever $w_1 \cdot p + w_2 \cdot h < w_1 \cdot \mathbf{p} + w_2 \cdot \mathbf{h}$.

Thus a key algorithmic question in ERCI is how to efficiently explore the Pareto front $\mathcal{F}_{\mathbb{S}}$.

C. Regret-Based ERCI

To algorithmically explore the Pareto-curve, we re-parameterize the ERCI problem.

First, we find the two special points induced by (1) optimizing performance and only then randomization (the topmost green point in the figures) and (2) optimizing randomization and only then performance (the rightmost green point). As we have seen, these restrict the domain in which we can actually trade performance for randomness. We define $h^* \stackrel{\text{def}}{=} \max\{h \mid \exists p \text{ s.t. } \langle p, h \rangle \in \mathbb{S}\}$, i.e., the largest randomness that can be guaranteed by any ego-policy. Likewise, we define $p^* \stackrel{\text{def}}{=} \max\{p \mid \exists h \text{ s.t. } \langle p, h \rangle \in \mathbb{S}\}$, i.e., the largest performance that can be guaranteed by any ego-policy. Then, we define $p^- \stackrel{\text{def}}{=} \max\{p \mid \langle p, h^* \rangle \in \mathbb{S}\}$, the best performance that ego can guarantee while guaranteeing optimal randomness. Likewise, we define the analogous $h^- \stackrel{\text{def}}{=} \max\{h \mid \langle p^*, h \rangle \in \mathbb{S}\}$. We thus obtain two points on the Pareto front: $\langle p^-, h^* \rangle$ and $\langle p^*, h^- \rangle$, and intuitively, we can trade between these two points following the Pareto front.

Now, rather that fixing $\mathbf{p}$ and $\mathbf{h}$ a priori, we seek to guarantee some percentage of the independently achievable soft constraint and causal entropy measure. We re-parameterize ERCI as follows:

$$\mathbf{p}_\epsilon \stackrel{\text{def}}{=} \epsilon \cdot (p^* - p^-) + p^- \quad \mathbf{h}_\delta \stackrel{\text{def}}{=} \delta \cdot (h^* - h^-) + h^- \tag{13}$$

where $\epsilon, \delta \in [0, 1]$. We call this version of ERCI *regret-based*. We remark that the reparameterization is not only beneficial from a usability point-of-view, but it also eases our exposition. Geometrically, after computing p^* and h^*, we know that the left triangle in Fig. 4d is definitively realizable, and the regret-based ERCI asks whether the white circle is also realizable (where the point of the white point is given by ϵ and δ. Together, we obtain the following (core) ERCI problem.

> **The Core ERCI Problem**: Given an finite acyclic SG $\mathcal{G}$, with terminal states, $s_\top$ and $s_\bot$, and thresholds $\epsilon, \delta \in [0, 1]$, find a ego-policy σ_{ego} s.t. for every env-policy σ_{env}:
> 1) (*soft constraint*) $\Pr(\mathsf{last}(\xi) = s_\top \mid \sigma) \geq \mathbf{p}_\epsilon$
> 2) (*randomness constraint*) $H(\sigma) \geq \mathbf{h}_\delta$
>
> where $\sigma = \langle \sigma_{\text{ego}}, \sigma_{\text{env}} \rangle$.

Finally, it is helpful to think about the Pareto front as a function of randomization in this reparameterization. We define a characteristic function which given a target performance ratio, ϵ, yields the optimal randomness ratio, δ:

$$\begin{aligned} &f_{\mathbb{S}} \colon [0, 1] \to [0, 1] \\ &f_{\mathbb{S}}(\delta) = \max_{\epsilon}\{\mathbf{h}_\delta \mid \langle \mathbf{p}_\epsilon, \mathbf{h}_\delta \rangle \in \mathbb{S}\} \end{aligned} \tag{14}$$

Proposition 2. *$f_{\mathbb{S}}$ is continuous and (strictly) decreasing.*

We shall temporarily postpone the proof of Prop. 2. For now, one case observe that (non-strict) monotone decreasing follows directly from convexity and using the adequate domains. Finally, the set $\mathbb{S}$ is (in general) *not* a finite polytope – the MDP in Fig. 3a serves as an example. Nevertheless, $\mathbb{S}$ can be well approximated with finitely many vertices, see Ex. 8.

With these facts, we are now well-equipped to develop the algorithms in Sec. V for MDPs and Sec. VI for SGs.

V. The Control Improvisation Problem for MDPs

We present an algorithm for the control improvisation problem for MDPs, which in the next section, will serve as a subroutine for an algorithm on SGs. Our goal shall be to instantiate the approximation scheme from the previous section. In particular, we seek to find points on the Pareto curve $\mathcal{F}_{\mathbb{S}}$ and incrementally build up $\mathcal{F} \subseteq \mathcal{F}_{\mathbb{S}}$.

A. Rationality

To start, recall that an MDP is a stochastic game with no action choices for the environment, i.e., the environment is purely stochastic and the only degree of freedom is ego's policy. The key idea for finding points on the Pareto-curve is to rephrase the trade-off between randomization and performance as a degree in rationality λ of the policy. Formally, the rationality corresponds to the following scalarization of our multi-objective problem [37],

$$J_\lambda(\sigma) \stackrel{\text{def}}{=} \Big\langle 1, \lambda \Big\rangle \cdot \Big\langle h_\sigma, p_\sigma \Big\rangle. \tag{15}$$

In context of MDPs, the **unique** (ego-)policy that optimizes (15) is given by a smooth variant of the Bellman equations [60, 57]. Namely, let smax denote the log-sum-exp operator, i.e., $\text{smax}(X) \stackrel{\text{def}}{=} \log\left(\sum_{x\in X} e^x\right)$. For each rationality $\lambda \in [0, \infty)$, we define a policy σ_λ – using $s = \mathsf{last}(\xi)$ – as follows:

$$\sigma_\lambda(\alpha \mid s) \stackrel{\text{def}}{=} \exp(Q_\lambda(s, \alpha) - V_\lambda(s)) \tag{16}$$

$$V_\lambda(s) \stackrel{\text{def}}{=} \begin{cases} \lambda \cdot [s = s_\top] & \text{if } s \in \{s_\top, s_\bot\}, \\ \text{smax}_{\alpha\in A(s)} Q_\lambda(s, \alpha) & \text{otherwise.} \end{cases} \tag{17}$$

$$Q_\lambda(s, \alpha) \stackrel{\text{def}}{=} \sum_{s'} P(s, \alpha, s') \cdot V_\lambda(s'). \tag{18}$$

To ease notation, we denote $x_\lambda \stackrel{\text{def}}{=} x_{\sigma_\lambda}, p_\lambda \stackrel{\text{def}}{=} p_{\sigma_\lambda}, h_\lambda \stackrel{\text{def}}{=} h_{\sigma_\lambda}$. Intuitively, as $\lambda \to 0$, σ_λ approaches the uniform distribution over *all available actions*. Note that this policy maximizes (causal) entropy, and thus $h^* = h_0$. As $\lambda \to \infty$, this variant of the Bellman equations coincides with the standard Bellman equations [47], where σ_λ selects (uniformly) from actions *that maximize performance*. Furthermore, the monotonicity and smoothness of the above Bellman equations yields the following proposition.

Proposition 3. *p_λ is continuously (and strictly) increasing in λ and h_λ is smoothly (and strictly) decreasing in λ.*

In terms of $f_{\mathbb{S}}$, we can define:

$$\epsilon_\lambda \stackrel{\text{def}}{=} \frac{p_\lambda - p_0}{p_\infty} + p_0 \quad \text{and} \quad \delta_\lambda \stackrel{\text{def}}{=} \frac{h_\lambda - h_\infty}{h_0} + h_\infty. \tag{19}$$

Then, because σ_λ maximizes randomness given a target performance, one derives:

$$f_{\mathbb{S}}(\delta_\lambda) = \epsilon_\lambda. \tag{20}$$

What remains is to instantiate the approximation scheme for the Pareto front by varying the optimization direction $\langle \lambda, 1\rangle$.[6] In particular, we construct $\mathcal{F} = \{x_\lambda \mid \lambda \in \{\lambda_1, \lambda_2, \ldots\}\}$ until $\mathcal{F}$ contains a witness to either realizability or unrealizability of the ERCI instance. We notice that the scalarization in (15) means that we may additionally exploit witnesses to unrealizability as outlined in Remark 2. In the remainder of this section, we improve upon randomly selecting values for λ.

B. Targeted Pareto-exploration

The key ingredient to improve upon arbitrarily selecting $\lambda_1, \ldots \lambda_i$ is to exploit additional structure of the rationality.

We propose a three staged sequence: (i) Compute x_λ for the end points $\lambda \in \{0, \infty\}$. (ii) Double λ (starting at $\lambda = 1$) until $h_\lambda \le \mathbf{h}$, yielding $\lambda_1 \ldots \lambda_j$. (iii) Binary search for $\lambda \in [\lambda_{j-1}, \lambda_j]$. We illustrate the idea in Fig. 4e.

The algorithm terminates almost surely, that is: the algorithm halts if $\langle \mathbf{p}, \mathbf{h}\rangle$ is not on $\mathcal{F}_{\mathbb{S}}$ (or if we happen to exactly hit $\langle \mathbf{p}, \mathbf{h}\rangle$ by selecting some rationality λ). As the Pareto front has measure 0, we argue that not halting is thus merely a technical concern, as a small perturbation to the ERCI instance (i.e. a *smoothed analysis* [53]) on $\mathcal{G}$ admits decidability.

[6] Assuming $p^*, h^* \neq 0$ (which would otherwise yield trivial $\mathbb{S}$ and $\mathcal{F}_{\mathbb{S}}$)

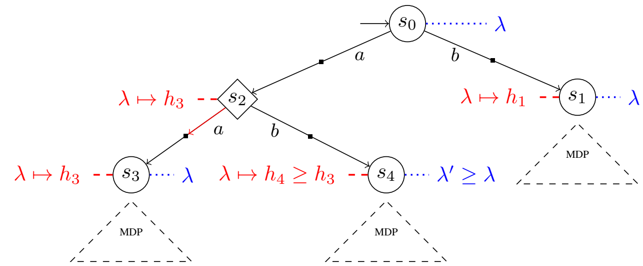


Fig. 5. SG to illustrate entropy matching policies.

> Our approximation scheme yields a semi-decision process which halts iff either (a) $\langle \mathbf{p}, \mathbf{h}\rangle$ is bounded away from $\mathcal{F}_{\mathbb{S}}$ *or* (b) $\langle \mathbf{p}, \mathbf{h}\rangle$ is dominated by x_{λ_i}.

Next, observe that if we terminate the binary search when the search region is smaller than Δ, this approximation scheme becomes linear in the MDP size and logarithmic in the final rationality, λ_*, and the resolution, Δ, i.e., the run-time is,

$$\mathcal{O}\Big(\underbrace{|\mathcal{G}|}_{\text{Evaluate } x_\lambda} \cdot \overbrace{\log(\lambda_*)}^{\text{Doubling Phase}} \cdot \underbrace{\log(1/\Delta)}_{\text{Binary Search}}\Big) \tag{21}$$

Finally, before generalizing to stochastic games, we observe that in practice, $\lambda = 100$ yields a nearly optimal policy, and thus one can often assume $\lambda_* \le 100$ in our run-time analysis.

VI. The Control Improvisation Problem for SGs

MDP algorithm in hand, we are now ready to provide an algorithm for stochastic games.

Environment Policies. We begin with three observations about the env-policies. First, for ERCI, we can assume an adversary for env that aims to foil ego achieving both the performance *and* randomization requirement. We call such a env-policy *violating*. For a policy to be violating, it suffices to violate, against every ego-policy independently, either performance *or* randomization. Second, if there is a violating env-policy, there is a deterministic env-policy that proves this. In particular, at every state, σ_{env} may choose to violate either constraint via the appropriate action with no incentive to randomize. Third, fixing an environment policy reduces $\mathcal{G}$ to a MDP $\mathcal{G}[\sigma_{\mathsf{env}}]$.

A Sufficient Class of Policies. For MDPs, we have seen that varying rationality is sufficient to explore the Pareto curve. We show that we can adapt that idea to a class we call *entropy matching policies*, which may be indexed by the (initial) rationality. In the initial state, we start by assuming that env selects a (deterministic) policy, $\sigma^\lambda_{\mathsf{env}}$, that lexicographically minimizes the guaranteed randomness, followed by performance. On the sub-graph, $\mathcal{G}[\sigma^\lambda_{\mathsf{env}}]$, ego employs the corresponding entropy maximizing policy for the MDP $\mathcal{G}[\sigma^\lambda_{\mathsf{env}}]$. Whenever env diverges from the entropy minimizing policy (to decrease the induced performance), we let ego increase its rationality such that it still induces the same guaranteed randomness. We refer to this idea as *entropy matching*. The idea is that the rationality at the initial state induces a worst-case entropy, and whatever env chooses to do, throughout the SG, we ensure that we indeed obtain this entropy. The policy thus tracks this entropy and if necessary adapts the rationality (which we call *replanning*). Replanning ensures we obtain the optimal performance from a particular point while still ensuring the required randomness.

Example 9. We sketch an entropy matching policy in Fig. 5. In particular, we show part of a SG. For some fixed rationality λ, we annotate in red, on the left of the SG states, the entropy obtained when assuming that env plays an entropy-minimizing policy as outlined above. In particular, this means that in s_2, env selects action a. Now, our entropy-matching policy (in blue, on the right) will play with rationality λ, unless state s_4 is reached. As this ensures a higher entropy, we may now select a higher rationality, λ'.

Soundness and Completeness. Importantly, observe that because fixing a policy for ego yields a verifiable point in $\mathbb{S}$, any witness for realizability we find is trivially sound. For completeness, we can restrict ourselves to the case in which our algorithm claims the ERCI instance unrealizable. Surprisingly, the class of policies we consider suffices, and the algorithm is thus sound and (whenever halting) complete (proof provided in Sec X). That is, all guaranteed points are witnessed by an entropy matching policy!

Further, observe that as a corollary of the entropy matching family being complete, it must be the case that $f_{\mathbb{S}}(h_\lambda)$ inherits continuity and (strict) monotonicity from the MDP case. Namely, at each env state, the achievable points $\mathbb{S}$ are necessarily the intersection of the achievable points of the sub-graphs. By induction, (with the MDP base case), we obtain continuity and strict monotonicity.

Algorithm: Memoizing Pareto Fronts. We propose approximating the Pareto front using the same three staged sequence of exploring rationality coefficients (at the initial state) as the MDP case: (1) endpoints, (2) doubling, (3) binary search.

To perform the above computations efficiently, we adopt a geometric perspective. Namely, observe that each node of $\mathcal{G}$ indexes a sub-graph, which has a corresponding Pareto front for trading performance for randomness. Further, note that the Pareto front at an env node is the intersection of the Pareto fronts of its child nodes. Entropy matching corresponds to "switching" between Pareto fronts and adjusting the optimization direction by increasing the rationality. Thus, by traversing the graph from the terminal states to the initial state, approximating Pareto fronts along the way, one can memoize how to trade performance for randomness at any given node. This preprocessing enables determining the minimum entropy response for any optimization direction and quickly replanning via a convex combination of Pareto optimal policies.

Approximate Pareto Fronts. Of course, by varying λ, one can only construct approximate Pareto fronts $\hat{\mathcal{F}} \subseteq \mathcal{F}_{\mathbb{S}}$. We propose the following high-level algorithm to adapt the above algorithm to the case where each Pareto front approximation introduces at most κ error along the performance axis.

1) Let τ denote the length of the longest path in $\mathcal{G}$.
2) Let $0 < \kappa < 1$ be some arbitrary initial tolerance.
3) Recursively compute κ-close Pareto fronts for each successor state using replanning.
4) If the any minimum entropy action cannot be determined or $\mathbf{p}$ is within $\kappa \cdot \tau$ distance to (but outside of) $\hat{\mathcal{F}}$, halve κ and repeat.
5) Otherwise, perform the entropy matching algorithm (with initial entropy $\mathbf{h}$) using these Pareto fronts and return the resulting policy (if on exists).

The soundness of this algorithm relies on the following critical facts: (1) Given sufficient resolution, the minimum entropy env-actions can be determined. (2) The resulting entropy depends solely on the resulting sub-graph (and is independent of the current Pareto approximation). (3) Thus, when querying points on $\mathcal{F}_{\mathbb{S}}$, error can only accumulate for p. (4) Next, observe that p is computed using convex combinations of entropy matched points on Pareto approximations. (5) Convex combinations of an error interval cannot increase the error, i.e.,

$$q \cdot [x, x+\kappa] + \bar{q} \cdot [y, y+\kappa] = [z, z+\kappa], \tag{22}$$

where $z = q \cdot x + \bar{q} \cdot y$. Thus, so long as $\kappa \cdot \tau$ is enough resolution to answer $p_\lambda < \mathbf{p}$, one obtains a semi-decision procedure as in the MDP case.

Termination and Run Time. First, as in the MDP case, the algorithm terminates almost surely, with the exception occurring only for a subset of the Pareto front. Below, we give an output-sensitive analysis of the run time (assuming it does halt). If κ^* tolerance is required to terminate, then the κ search introduces $\mathcal{O}(\log(1/\kappa^*))$ iterations. Next, observe that each node need process a given rationality coefficient at most once. Further, looking up which pair of rationalities are need to upper and lower bound the performance for a given randomness can be done in logarithmic time via binary search on rationality coefficients. As the corresponding bounds and convex combinations can be computed in constant time, this means this algorithm runs in time:

$$\mathcal{O}\Big(\log(1/\kappa^*) \cdot N_\lambda \cdot \log(N_\lambda) \cdot |\mathcal{G}|\Big), \tag{23}$$

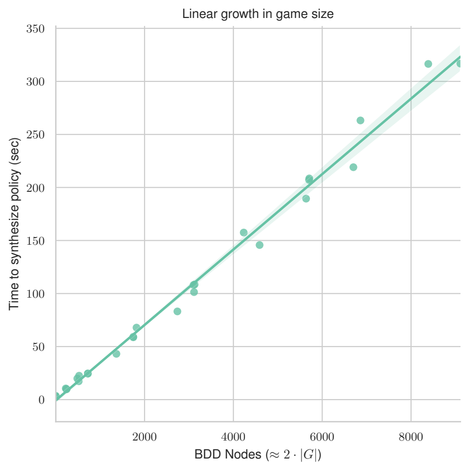
where, N_λ is the number of unique rationality coefficients processed. If, as in the MDP case, one assumes a maximum rationality coefficient λ^* and a minimum rationality resolution Δ, one obtains:

$$\mathcal{O}\Big(\underbrace{\log(1/\kappa^*)}_{\kappa \text{ search}} \cdot \underbrace{\lambda^*/\Delta \cdot \overbrace{\log(\lambda^*/\Delta)}^{\text{Replanning}} \cdot |\mathcal{G}|}_{\text{Evaluate } \lambda}\Big). \tag{24}$$

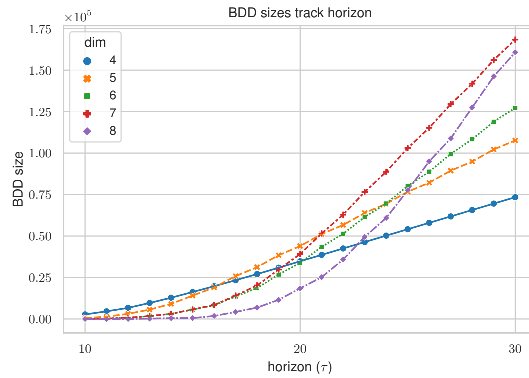
The above however is very conservative and empirically we observe N_λ bounded far away from λ^*/Δ.

VII. Implementation and Empirical Evaluation

Setup. To experimentally validate the feasibility of our ERCI algorithm for SGs, we implemented [56] our algorithm in Python. Inspired by the recent work on compressing MDPs



(a) Experimental times for computing Pareto front of a variety delivery drone problems.



(b) BDD graph size as a function of horizon for the problems in our benchmark suite. Distribution of problems, non-uniform in horizon to avoid small horizon artifacts.

Fig. 6. Plots to illustrate scalability

for specification inference [57], each SG was represented as a Binary Decision Diagram (BDD) [9] using the `dd` and `py-aiger` python packages [38, 55].

We investigate the motivating example. Specifically, our experiments used a $k \times k$ grid discretization of the workspace (cf. Fig. 1), for $k \in \{4, 5, 6, 7\}$ where the four target houses lie in $\{\lfloor k/3 \rfloor, \lfloor 2k/3 \rfloor\}^2$, and the drones D_{test} and D_{new} are initially at in the bottom left corner and top right house resp. Furthermore, for simplicity, we embedded the avoid crash condition as part of the soft constraint, rather than a hard constraint[7]. We took ego's dynamics to be deterministic and modeled env as visiting each house in either clock-wise or counter-wise order, where the orientation can switch with (a potentially state dependent) probability $p \in [1/100, 1/50]$ whenever a house is visited. Next, we considered an alternation between ego and env to be a single logical time step, and (non-uniformly) instantiate problem instances with horizons ranging from 6 to 18, i.e., paths ranged from length 12 to 36.

Results. First and foremost, we succeed in synthesizing controllers in the mentioned setup. The controller randomizes its behavior while meeting the specification, which is not surprising as the algorithm yields a correct-by-construction policy.

Next, we consider the practical run time of our algorithm. As Fig 6a demonstrates, the empirical time to estimate the Pareto front seemed to increase linearly with our SG encoding – which is consistent with our complexity analysis. Moreover, our encoding seems to linearly track with the horizon for all k (Fig. 6b), suggesting that the overall run time grows linearly in the horizon within our parameterization. When combined with the potential to parallelize across the rationality coefficients, these results suggest that practical optimizations to our ERCI algorithm may admit usage on other more complicated benchmarks. Finally, we remark that the use of a decision diagram encoding did indeed dramatically decrease the size of the SG (with negligible overhead).[8]

VIII. Discussion and Related Work

A. Control Improvisation in the Literature

In this section, we briefly compare ERCI with other forms of control improvisation. Firstly, we observe that general Control Improvisation has been proposed in stochastic environments for lane changing [22] and imitating power usage in households [2]. However, in those both settings, the randomness constraint is phrased as an upper-bound on the probability of indefinitely-long paths. Consequently, those randomness constraints are trivially satisfied. In comparison, we consider the synthesis of policies that necessarily randomize in presence of stochastic behavior in the environment. The closest prior work is to ours is Reactive Control Improvisation (RCI) for (deterministic) 2-player games [19]. As in ERCI, RCI features three kinds of constraints; hard, soft, and randomness. As in ERCI, RCI can be preprocessed resulting in the following core problem.

> **The Core RCI Problem**: Given a finite acyclic (deterministic) SG $\mathcal{G}$, with terminal states, $s_\top$ and $s_\bot$, and thresholds $\mathbf{p} \in (0, 1)$ and $\mathbf{h} \in [0, \infty)$, find a ego-policy σ_{ego} such that for every env-policy σ_{env}
> 1) (*soft constraint*) $\Pr(\mathsf{last}(\xi) = s_\top \mid \sigma) \geq \mathbf{p}$,
> 2) (*randomness*) $\max_\xi \Pr(\xi \mid \sigma) \leq \mathbf{d}$,
>
> where $\sigma = \langle \sigma_{\text{ego}}, \sigma_{\text{env}} \rangle$.

While RCI is only applied to deterministic SGs in [19], there is nothing in the definition that prevents its application to the general class of SGs.[9] We observe that then, the only difference

[7] Note that counter-intuitively, only using soft constraints generally results in harder instances as the compressed SGs are larger.

[8] For example, the $(k = 8, \text{horizon} = 18)$ case is encoded using a 505,100 node BDD ($|\mathcal{G}| = 6,861$ nodes). Compare with the direct encoding $|\mathcal{G}| = |S| \cdot \tau \cdot |\text{monitor state}| = (8 \times 8)^2 \cdot (2 \cdot 18) \cdot (2^4 \cdot 2) \approx 37,000$.

[9] However, this does not mean that the algorithm to compute a solution carries over to the general case

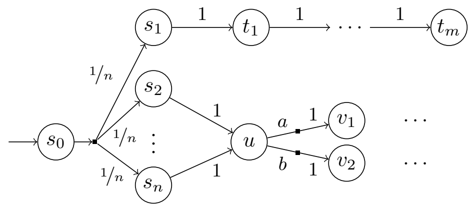


Fig. 7. Example Illustrating the problem with RCI in stochastic environments.

between ERCI and RCI is that we use causal entropy rather than an upper bound on the probability of a path to enforce randomness. Below we address two problems with bounding the maximum probability of a trace.

First, RCI fails to account for causality when measuring randomness. In deterministic systems, for which RCI was conceived, this distinction is unnecessary, but stochastic systems must deal with counter-factuals. In practice, RCI encodes an agent model that is systematically overly optimistic regarding the outcomes of dynamics transitions [37]. This results in policies with worse performance given a fixed randomness target. In the context of our motivating drone example, applying RCI thus results in a policy that is both quantitatively and qualitatively less random than the ERCI.

Second, RCI fails to enforce randomization if there exists *any* path with sufficiently high probability. The next (pathological) example illustrates.

Example 10. Consider the SG (actually, an MDP where we omit the env-states) in Fig. 7. First consider that under each scheduler, the path from s_0 to t_m has probability $1/n$. In particular, this means that a feasible RCI instance (applied to an SG) must have $\mathbf{d} \geq 1/n$. At the same time, every path in the SG already has probability at most $1/n$, and thus, every scheduler that satisfies the randomness constraint for $\delta = 1$ satisfies it for any $\mathbf{d} \geq 1/n$. Thus, for this MDP, the RCI formulation fails to enforce any randomization in the ego-policy. By contrast, a causal entropy constraint from ERCI will continuously trade-off randomness for performance.

On the other hand, one can observe that in reality, proposed algorithms for solving RCI equally distribute probability mass across the maximum number of paths that ego can guarantee [19]. We remark that because (1) causal entropy reduces to non-causal entropy in deterministic dynamics and (2) uniform distributions maximize entropy, our proposed entropy matching family exactly agrees with existing RCI algorithms on deterministic SGs. Thus, we observe the following proposition.

Proposition 4. *There exists a computable function,*

$$f\colon (\mathbf{d}, \mathcal{G}) \mapsto \mathbf{h},$$

such that, for any deterministic SG, $\mathcal{G}$, and performance threshold $\mathbf{p}$, there exists an ego*-policy solving the RCI problem with threshold $\mathbf{d}$ iff there exists a* ego*-policy solving the ERCI problem with threshold $\mathbf{h} = f(\mathbf{d}, \mathcal{G})$.*

B. Additional Related Work

Synthesis in MDPs with multiple hard and soft constraints (often over indefinite horizons) is a well-studied problem [11, 16, 18, 49]. In this setting, one generates deterministic policies and their convex combinations. Put differently, some degree of randomization is *not an objective*, but rather a consequence. Interestingly, in [15] the optimal policies in *absence* of randomization are investigated. Along similar lines, [8] trades average performance for less variance, thereby implicitly trading off the average and the worst-case performance. The original results sparked interest in different extension to MDPs and the type of soft constraints, such as continuous MDPs [25] and continuous-time MDPs [48], cost-bounded reachability [26], or mean-payoff properties [7]. The algorithms have also been extended towards stochastic games [13, 35]. Finally, notions of lexicographic multi-objective synthesis [12] – in which one optimizes a secondary criterion among all policies that are optimal with respect to a first criterion bare some resemblance with the algorithm we consider. The aforementioned algorithms have been put in a robotics context in [36]. Finding policies that optimize reward objectives is well-studied in the field of reinforcement learning, and has been extended to generate Pareto fronts for multiple objectives [41, 44].

Next, our core ERCI instance can be seen as a multi-objective path problem [4, 42, 59]. The literature on multi-object path finding differs prominently from ERCI in two aspects: they do not trade-off randomization and performance, and they do not trade-off declarative and formal constraints with the accompanying formal guarentees, but are more search-based.

Another related domain is the problem of (randomly) patrolling a perimeters and points of interest [1, 5, 46]. Closest to our work are formalisms rooted in game-theory, such as *Stackelberg games* [51, 45]. Stackelberg games have been extending to Stackelberg planning [52] in which a trade-off between the cost for the defender and the attacker can be investigated. Most related are the zero-sum *patrolling games* introduced in [3], which has led to numerous practical solutions [54]. Patrolling games are explicitly games between an intruder and a defender, and there is no stochastic environment. Adding additional objectives makes solving these problems harder [34] and in general, the obtained policies are no longer applicable. To overcome this, a specific set of fixed objectives has been added to these games recently [34]. The large common aspect in all of this work is that optimal strategies do randomize. As in the synthesis work above, this is a consequence of the objectives rather than an objective in itself. In comparison, we provide a general framework and in particular support stochastic environments.

Finally, entropy as an optimization objective for MDPs with fixed rewards has been well studied [50], particularly in the context of regularizing (robustifying) inverse and reinforcement learning [60, 23]. The primary distinction from our work (in the MDP setting) is the unspecified (performance/entropy) trade-off. Nevertheless, as previously discussed, the specification varient of this literature served as the basis for our MDP subroutine [57]. Beyond Markov models, the (uniform) randomization over languages in finite automata [29, 32] or over propositional formulae [31, 6, 10] has received quite some attention, however neither of those approaches support the notion of soft constraints or the related trade-offs.

IX. Conclusion

This paper presented ERCI, a framework to control improvisation in stochastic games. Our results show that ERCI can be used to synthesize policies that besides meeting temporal logic specifications induce varying behavior, e.g., to test and certify the correctness of other robots. Future work includes applying the framework to a broader spectrum of applications and extending the theory to games with imperfect information.

Acknowledgments: This work is partially supported by NSF grants 1545126 (VeHICaL), 1646208 and 1837132, by the DARPA contracts FA8750-18-C-0101 (Assured Autonomy) and FA8750-20-C-0156 (SDCPS), by Berkeley Deep Drive, and by Toyota under the iCyPhy center.

References

[1] Noa Agmon, Sarit Kraus, and Gal A. Kaminka. Multi-robot perimeter patrol in adversarial settings. In *ICRA*, pages 2339–2345. IEEE, 2008.

[2] Ilge Akkaya, Daniel J. Fremont, Rafael Valle, Alexandre Donzé, Edward A. Lee, and Sanjit A. Seshia. Control improvisation with probabilistic temporal specifications. In *IoTDI*, pages 187–198. IEEE Computer Society, 2016.

[3] Steve Alpern, Alec Morton, and Katerina Papadaki. Patrolling games. *Oper. Res.*, 59(5):1246–1257, 2011.

[4] Francesco Amigoni and Alessandro Gallo. A multi-objective exploration strategy for mobile robots. In *ICRA*, pages 3850–3855. IEEE, 2005.

[5] Francesco Amigoni, Nicola Basilico, and Nicola Gatti. Finding the optimal strategies for robotic patrolling with adversaries in topologically-represented environments. In *ICRA*, pages 819–824. IEEE, 2009.

[6] Mihir Bellare, Oded Goldreich, and Erez Petrank. Uniform generation of np-witnesses using an np-oracle. *Inf. Comput.*, 163(2):510–526, 2000.

[7] Tomás Brázdil, Václav Brozek, Krishnendu Chatterjee, Vojtech Forejt, and Antonín Kucera. Two views on multiple mean-payoff objectives in Markov decision processes. *Log. Methods Comput. Sci.*, 10(1), 2014.

[8] Tomás Brázdil, Krishnendu Chatterjee, Vojtech Forejt, and Antonín Kucera. Trading performance for stability in Markov decision processes. *J. Comput. Syst. Sci.*, 84: 144–170, 2017.

[9] Randal E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv.*, 24(3):293–318, 1992.

[10] Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. Balancing scalability and uniformity in SAT witness generator. In *DAC*, pages 60:1–60:6. ACM, 2014.

[11] Krishnendu Chatterjee, Rupak Majumdar, and Thomas A. Henzinger. Markov decision processes with multiple objectives. In *STACS*, volume 3884 of *LNCS*, pages 325–336. Springer, 2006.

[12] Krishnendu Chatterjee, Joost-Pieter Katoen, Maximilian Weininger, and Tobias Winkler. Stochastic games with lexicographic reachability-safety objectives. In *CAV (2)*, volume 12225 of *LNCS*, pages 398–420. Springer, 2020.

[13] Taolue Chen, Vojtech Forejt, Marta Z. Kwiatkowska, Aistis Simaitis, and Clemens Wiltsche. On stochastic games with multiple objectives. In *MFCS*, volume 8087 of *LNCS*, pages 266–277. Springer, 2013.

[14] Anne Condon. On algorithms for simple stochastic games. In *Advances In Computational Complexity Theory*, volume 13 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 51–71. DIMACS/AMS, 1990.

[15] Florent Delgrange, Joost-Pieter Katoen, Tim Quatmann, and Mickael Randour. Simple strategies in multi-objective MDPs. In *TACAS (1)*, volume 12078 of *LNCS*, pages 346–364. Springer, 2020.

[16] Kousha Etessami, Marta Z. Kwiatkowska, Moshe Y. Vardi, and Mihalis Yannakakis. Multi-objective model checking of Markov decision processes. In *TACAS*, volume 4424 of *LNCS*, pages 50–65. Springer, 2007.

[17] Benjamin Eysenbach and Sergey Levine. If maxent RL is the answer, what is the question? *CoRR*, abs/1910.01913, 2019.

[18] Vojtech Forejt, Marta Z. Kwiatkowska, and David Parker. Pareto curves for probabilistic model checking. In *ATVA*, volume 7561 of *LNCS*, pages 317–332. Springer, 2012.

[19] Daniel J. Fremont and Sanjit A. Seshia. Reactive control improvisation. In *CAV (1)*, volume 10981 of *LNCS*, pages 307–326. Springer, 2018.

[20] Daniel J. Fremont, Alexandre Donzé, Sanjit A. Seshia, and David Wessel. Control improvisation. In *FSTTCS*, volume 45 of *LIPIcs*, pages 463–474. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[21] Jie Fu, Nikolay Atanasov, Ufuk Topcu, and George J. Pappas. Optimal temporal logic planning in probabilistic semantic maps. In *ICRA*, pages 3690–3697. IEEE, 2016.

[22] Jin I. Ge and Richard M. Murray. Voluntary lane-change policy synthesis with control improvisation. In *CDC*, pages 3640–3647. IEEE, 2018.

[23] Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized Markov decision processes. In *ICML*, volume 97 of *PMLR*, pages 2160–2169. PMLR, 2019.

[24] Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, pages 854–860. IJCAI/AAAI, 2013.

[25] Sofie Haesaert, Petter Nilsson, and Sadegh Soudjani. Formal multi-objective synthesis of continuous-state MDPs. *IEEE Control. Syst. Lett.*, 5(5):1765–1770, 2021.

[26] Arnd Hartmanns, Sebastian Junges, Joost-Pieter Katoen, and Tim Quatmann. Multi-cost bounded tradeoff analysis in MDP. *J. Autom. Reason.*, 64(7):1483–1522, 2020.

[27] Keliang He, Morteza Lahijanian, Lydia E. Kavraki, and

Moshe Y. Vardi. Reactive synthesis for finite tasks under resource constraints. In *IROS*, pages 5326–5332. IEEE, 2017.

[28] Keliang He, Andrew M. Wells, Lydia E. Kavraki, and Moshe Y. Vardi. Efficient symbolic reactive synthesis for finite-horizon tasks. In *ICRA*, pages 8993–8999. IEEE, 2019.

[29] Timothy J. Hickey and Jacques Cohen. Uniform random generation of strings in a context-free language. *SIAM J. Comput.*, 12(4):645–655, 1983.

[30] Matanya B. Horowitz, Eric M. Wolff, and Richard M. Murray. A compositional approach to stochastic optimal control with co-safe temporal logic specifications. In *IROS*, pages 1466–1473. IEEE, 2014.

[31] Mark Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986.

[32] Sampath Kannan, Z. Sweedyk, and Stephen R. Mahaney. Counting and random generation of strings in regular languages. In *SODA*, pages 551–557. ACM/SIAM, 1995.

[33] Yiannis Kantaros, Matthew Malencia, Vijay Kumar, and George J. Pappas. Reactive temporal logic planning for multiple robots in unknown environments. In *ICRA*, pages 11479–11485. IEEE, 2020.

[34] David Klaska, Antonín Kucera, and Vojtech Rehák. Adversarial patrolling with drones. In *AAMAS*, pages 629–637. IFAAMAS, 2020.

[35] Marta Kwiatkowska, David Parker, and Clemens Wiltsche. Prism-games: verification and strategy synthesis for stochastic multi-player games with multiple objectives. *Int. J. Softw. Tools Technol. Transf.*, 20(2):195–210, 2018.

[36] Bruno Lacerda, Fatma Faruq, David Parker, and Nick Hawes. Probabilistic planning with formal performance guarantees for mobile service robots. *Int. J. Robotics Res.*, 38(9), 2019.

[37] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *CoRR*, abs/1805.00909, 2018. URL http://arxiv.org/abs/1805.00909.

[38] Scott C. Livingston. Binary Decision Diagrams (BDDs) in pure Python and Cython wrappers of CUDD, Sylvan, and BuDDy.

[39] James Massey. Causality, feedback and directed information. In *ISITA*, pages 303–305, 1990.

[40] Salar Moarref and Hadas Kress-Gazit. Automated synthesis of decentralized controllers for robot swarms from high-level temporal logic specifications. *Auton. Robots*, 44(3-4):585–600, 2020.

[41] Sriraam Natarajan and Prasad Tadepalli. Dynamic preferences in multi-criteria reinforcement learning. In *ICML*, volume 119 of *ACM International Conference Proceeding Series*, pages 601–608. ACM, 2005.

[42] Milad Nazarahari, Esmaeel Khanmirza, and Samira Doostie. Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. *Expert Syst. Appl.*, 115:106–120, 2019.

[43] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *ICML*, pages 663–670. Morgan Kaufmann, 2000.

[44] Simone Parisi, Matteo Pirotta, Nicola Smacchia, Luca Bascetta, and Marcello Restelli. Policy gradient approaches for multi-objective sequential decision making: A comparison. In *ADPRL*, pages 1–8. IEEE, 2014.

[45] Praveen Paruchuri, Jonathan P. Pearce, Milind Tambe, Fernando Ordóñez, and Sarit Kraus. An efficient heuristic approach for security against multiple adversaries. In *AAMAS*, page 181. IFAAMAS, 2007.

[46] David Portugal, Charles Pippin, Rui P. Rocha, and Henrik I. Christensen. Finding optimal routes for multi-robot patrolling in generic graphs. In *IROS*, pages 363–369. IEEE, 2014.

[47] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994.

[48] Tim Quatmann, Sebastian Junges, and Joost-Pieter Katoen. Markov automata with multiple objectives. In *CAV (1)*, volume 10426 of *LNCS*, pages 140–159. Springer, 2017.

[49] Mickael Randour, Jean-François Raskin, and Ocan Sankur. Percentile queries in multi-dimensional Markov decision processes. *Formal Methods Syst. Des.*, 50(2-3):207–248, 2017.

[50] Yagiz Savas, Melkior Ornik, Murat Cubuktepe, Mustafa O. Karabag, and Ufuk Topcu. Entropy maximization for Markov decision processes under temporal logic constraints. *IEEE Trans. Autom. Control.*, 65(4):1552–1567, 2020.

[51] Marwaan Simaan and Jose B Cruz. On the stackelberg strategy in nonzero-sum games. *Journal of Optimization Theory and Applications*, 11(5):533–555, 1973.

[52] Patrick Speicher, Marcel Steinmetz, Michael Backes, Jörg Hoffmann, and Robert Künnemann. Stackelberg planning: Towards effective leader-follower state space search. In *AAAI*, pages 6286–6293. AAAI Press, 2018.

[53] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis: an attempt to explain the behavior of algorithms in practice. *Commun. ACM*, 52(10):76–84, 2009.

[54] Milind Tambe. *Security and Game Theory - Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 2012.

[55] Marcell Vazquez-Chanlatte. mvcisback/py-aiger, 2018. URL https://doi.org/10.5281/zenodo.1326224.

[56] Marcell Vazquez-Chanlatte. Improvisers: A python library for synthesizing entropic reactive control improvisers for stochastic games., 2021. URL https://github.com/mvcisback/improvisers.

[57] Marcell Vazquez-Chanlatte and Sanjit A. Seshia. Maximum causal entropy specification inference from demonstrations. In *CAV (2)*, volume 12225 of *LNCS*, pages 255–278. Springer, 2020.

[58] Kai Weng Wong, Rüdiger Ehlers, and Hadas Kress-Gazit. Correct high-level robot behavior in environments with

unexpected events. In *Robotics: Science and Systems*, 2014.

[59] Jie Xu, Yunsheng Tian, Pingchuan Ma, Daniela Rus, Shinjiro Sueda, and Wojciech Matusik. Prediction-guided multi-objective reinforcement learning for continuous robot control. In *ICML*, volume 119 of *PMLR*, pages 10607–10616. PMLR, 2020.

[60] Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, 2010.

X. Proofs

A. Convexity of ERCI solution set

Proof Sketch Prop 1: Recall that a set is convex, if it is closed under convex-combinations[10]. Consider two points $\langle p, h\rangle, \langle p', h'\rangle \in \mathbb{S}$ achieved by σ_{ego} and σ'_{ego} respectively. Consider the new policy, π, defined by employing σ_{ego} with probability q and σ'_{ego} with probability $\bar{q} \overset{\text{def}}{=} 1 - q$. Because each policy *guarantees* its corresponding performance, this new policy has performance at least $q \cdot p + \bar{q} \cdot p'$. Similarly, by viewing π as a random variable and applying chain rule yields,

$$\begin{aligned} H_\tau(\sigma) \geq\ & q \cdot H(\mathcal{A}^{\text{ego}}_{1:\tau'} \mid\mid \mathcal{S}_{1:\tau} \mid \pi = \sigma_{\text{ego}}) + \\ & \bar{q} \cdot H(\mathcal{A}^{\text{ego}}_{1:\tau'} \mid\mid \mathcal{S}_{1:\tau} \mid \pi = \sigma'_{\text{ego}}) \\ =\ & q \cdot h + \bar{q} \cdot h'. \end{aligned} \tag{25}$$

Thus, any convex combination of guaranteed points is guaranteed by a convex combination of the corresponding ego policies. ■

B. Completeness of Entropy Matching for SGs

Proof Sketch of SG Completeness: We prove the statement by induction over the (acyclic) SG. First, observe that on games with only terminal nodes, completeness follows directly. Next, suppose the entropy matching family is complete on all subgraphs of $\mathcal{G}$. To simplify our proof, observe that w.l.o.g., we can restrict our attention to ERCI instances on the Pareto front, $\langle \mathbf{p}, \mathbf{h}\rangle \in \mathcal{F}_{\mathbb{S}}$. Next, for the sake of contradiction, we shall assume that no entropy matching policy achieves $\langle \mathbf{p}, \mathbf{h}\rangle$, but σ^*_{ego} does:

$$\forall \sigma_{\text{ego}} \in \{\sigma^\lambda_{\text{ego}}\}_\lambda \ .\ x_{\sigma_{\text{ego}}} \prec \langle \mathbf{p}, \mathbf{h}\rangle \tag{26}$$

$$\exists \sigma^*_{\text{ego}} \notin \{\sigma^\lambda_{\text{ego}}\}_\lambda \ .\ \langle \mathbf{p}, \mathbf{h}\rangle \preceq x_{\sigma^*_{\text{ego}}}. \tag{27}$$

Indeed, we may reformulate (27) to

$$\exists \sigma^*_{\text{ego}} \notin \{\sigma^\lambda_{\text{ego}}\}_\lambda \ .\ \langle \mathbf{p}, \mathbf{h}\rangle = x_{\sigma^*_{\text{ego}}} \tag{28}$$

as we assumed that $\langle \mathbf{p}, \mathbf{h}\rangle$ is Pareto-optimal.

Note that because the entropy matching family contains the maximizers and minimizers of entropy ($\lambda = \infty$ and $\lambda = 0$ resp.), and because increasing rationality monotonically decreases entropy, there must exist some rationality, λ, such that $\sigma^\lambda_{\text{ego}}$ induces entropy $\mathbf{h}$:

$$h_{\sigma^\lambda_{\text{ego}}} = \mathbf{h} = h_{\sigma^*_{\text{ego}}}, \tag{29}$$

where the second equality follows from (28). Next, let $\sigma^\lambda_{\text{env}}$ denote the min-entropy env-policy given $\sigma^\lambda_{\text{ego}}$, i.e., the policy that minimizes entropy in $\mathcal{G}[\sigma^\lambda_{\text{ego}}]$. Because σ^*_{ego} witnesses $\langle \mathbf{p}, \mathbf{h}\rangle$, it must be the case that:

$$h_{\langle \sigma^*_{\text{ego}}, \sigma^\lambda_{\text{env}}\rangle} \geq \mathbf{h} \qquad \text{and} \qquad p_{\langle \sigma^*_{\text{ego}}, \sigma^\lambda_{\text{env}}\rangle} \geq \mathbf{p} \tag{30}$$

Recalling that for MDPs, the maximum entropy policies as defined in (16)–(18) are the unique maximizers of entropy (given p), it must be the case that:

$$\mathbf{h} = h_{\langle \sigma^\lambda_{\text{ego}} \sigma^\lambda_{\text{env}}\rangle} \geq h_{\langle \sigma^*_{\text{ego}}, \sigma^\lambda_{\text{env}}\rangle} \geq \mathbf{h}, \tag{31}$$

and thus,

$$h_{\langle \sigma^\lambda_{\text{ego}} \sigma^\lambda_{\text{env}}\rangle} = h_{\langle \sigma^*_{\text{ego}}, \sigma^\lambda_{\text{env}}\rangle}. \tag{32}$$

Thus, from uniqueness on MDPs, $\sigma^\lambda_{\text{ego}}$ and σ^*_{ego} must exactly match on $\mathcal{G}[\sigma^\lambda_{\text{env}}]$ and must differ on some other subgraph. Applying the inductive hypothesis, we know that the entropy matching family is complete on these subgraphs, and thus if σ^*_{ego} achieves a given $\langle \mathbf{p}, \mathbf{h}\rangle$ on this subgraph, there must be an entropy matching that does so as well. Thus,

$$x_{\sigma^*_{\text{ego}}} \preceq x_{\sigma_{\lambda^*}}, \tag{33}$$

contradicting assumptions (26) and (27). Thus, entropy matching must be complete. ■