Entropy-Guided Control Improvisation

Marcell Vazquez-Chanlatte*, Sebastian Jungeś*, 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, behavior prediction and inference [57], and creating controller harnesses for fuzz testing (see motivating example in Sec. 11). 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 been studied in nondeterministic 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...
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 \(D_{\text{new}}\). As part of the process, the agency runs \(D_{\text{new}}\) through a series of tests. For example, given a certain delivery route, the agency investigates whether \(D_{\text{new}}\) successfully delivers packages while avoiding other delivery drones. To execute this test, the agency decides to synthesize a controller for another delivery drone, \(D_{\text{test}}\), to test if \(D_{\text{new}}\) can be certified.

Concretely, suppose we command \(D_{\text{new}}\) to continuously visit four houses in some workspace. We illustrate such a scenario in Fig. 1 in which \(D_{\text{new}}\) and \(D_{\text{test}}\) are shown as black square and white circle drones respectively. For this test scenario, the regulatory agency, wishes to examine how \(D_{\text{new}}\) responds to delivering packages to the red houses in the presence of \(D_{\text{test}}\). In particular, it would like to let \(D_{\text{test}}\) also deliver packages while avoiding \(D_{\text{new}}\). Importantly, to properly exercise \(D_{\text{new}}\), \(D_{\text{test}}\) 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 \(D_{\text{test}}\):

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 \(D_{\text{new}}\)’s controller when synthesizing \(D_{\text{test}}\). We illustrate by examining three models. In all models, we capture the behaviors of \(D_{\text{new}}\) and \(D_{\text{test}}\). We focus on \(D_{\text{new}}\), but the ideas carry over to modeling the actuation of \(D_{\text{test}}\).

Nondeterministic Model. The simplest approach to modeling is not to make any assumptions about \(D_{\text{new}}\) 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 \(D_{\text{test}}\) is too pessimistic, which leads to a bad test set. First, if \(D_{\text{new}}\) is unrestrained, then \(D_{\text{test}}\)’s behavior is severely limited, as it must behave conservatively to avoid collisions under all possible motions by \(D_{\text{new}}\) (even very unlikely motions). This limitation restricts the variance of its behavior, and it will not test \(D_{\text{new}}\)’s true behavior. A purely non-deterministic model for \(D_{\text{new}}\) thus may not lead to the synthesis of adequate behavior for \(D_{\text{test}}\).

Stochastic Model. Rather than the pessimistic nondeterministic (or adversarial) assumption, we may collect data about \(D_{\text{new}}\) and construct a stochastic model, e.g., using inverse reinforcement learning [43]. Concretely (but simplified), after examining the data, one observes that \(D_{\text{new}}\) 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 \(D_{\text{new}}\) much more precisely, and allows for more targeted test by \(D_{\text{test}}\).

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 \(D_{\text{new}}\). 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 \(\varepsilon\). 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
cases, the first player controls the behavior of \( D_{\text{test}} \) 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 \( \mathcal{G} = (S, t, A, P) \). The finite set of states \( S = S_{\text{ego}} \cup S_{\text{env}} \) is partitioned into a set \( S_{\text{ego}} \) of (controlled) ego-states and a set \( S_{\text{env}} \) of (uncontrolled) env-states. \( t \in S_{\text{ego}} \) is the initial state, \( A \) is a finite set of actions, and \( P: S \times A \to \text{Distr}(S) \) 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: \( P_{\text{ego}}: S_{\text{ego}} \times A \to \text{Distr}(S_{\text{env}}), P_{\text{env}}: S_{\text{env}} \times A \to \text{Distr}(S_{\text{ego}}) \). We identify the available actions as \( A(s) \triangleq \{ \alpha \mid P(s, \alpha) \neq \bot \} \). 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 \( \alpha \) is the set of states that are reached from \( s \) within one step with a positive transition probability, i.e., \( \text{Succ}(s, \alpha) \triangleq \{ s' \mid P(s, \alpha)(s') > 0 \} \), and \( \text{Succ}(s) \triangleq \bigcup_{\alpha \in A(s)} \text{Succ}(s, \alpha) \).

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 \( \alpha \), 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, \( s \in S_{\text{env}} \), 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.

B. Paths and Path Properties

A finite path, \( \xi \), of length \( n \) is a sequence \( s_0 \overset{\alpha_0}{\rightarrow} s_1 \overset{\alpha_1}{\rightarrow} s_2 \rightarrow \ldots \rightarrow s_n \) 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 \( \xi \), with last(\( \xi \)). Further, note that ego states are even indexed and env states are odd indexed as we assume alternation. A path, \( \xi' = s'_0 \overset{\alpha'_0}{\rightarrow} \ldots \overset{\alpha'_{n'}}{\rightarrow} s'_{n'} \), is a prefix of \( \xi \), if for all \( i \leq |\xi'| \), \( s_i = s'_i \) and for all \( i < |\xi'| \), \( \alpha_i = \alpha'_i \). The set of all finite paths of length \( n \) is denoted \( \text{Paths}_3^n \), and \( \text{Paths}^0 = \bigcup_{n \in \mathbb{N}} \text{Paths}_3^n \). We omit \( \mathcal{G} \) whenever it is clear from the context. It is helpful to partition paths based on their last state: \( [\text{Paths}]_\text{ego} = \{ \xi \in \text{Paths} \mid \text{last}(\xi) \in S_{\text{ego}} \} \) and \( [\text{Paths}]_{\text{env}} = \text{Paths} \setminus [\text{Paths}]_\text{ego} \).

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

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\(^1\). A scheduler is a tuple of player policies \( \sigma = (\sigma_{\text{ego}}, \sigma_{\text{env}}) \) with \( \sigma_i : [\text{Paths}]_i \rightarrow \text{Distr}(A) \) such that \( \text{support}(\sigma_i(\xi)) \subseteq A(\text{last}(\xi)) \) for each \( \xi \), i.e., for every history, the policy sets a distribution over the enabled successor actions. For a given path, \( \xi \) and a policy \( \sigma \), we denote by \( \sigma_i(\alpha \mid \xi) \) the distribution of actions induced by \( \sigma_i \) given the path \( \xi \). To ease notation, we liberally use the notation \( \sigma : \text{Paths} \rightarrow \text{Distr}(A) \), where this function is given dependent on which player owns the last state.

Example 3. An example for an ego-policy \( \sigma_{\text{ego}} \) is given by,

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

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

\[
\Pr(s \mid \sigma) \triangleq 1 \\
\Pr(\xi \mid \sigma) = \Pr(\xi' \mid \sigma \cdot \sigma(\alpha \mid \xi') \cdot P(\text{last}(\xi'), \alpha)(s'))
\]

where \( \xi = \xi' \overset{\alpha}{\rightarrow} s' \). The probability of a prefix-free set \( X \subseteq \text{Paths} \) 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 \( \tau = 20 \)” Technically, we represent these path properties as prefix-free sets of finite paths, \( \varphi \), reflecting some formal property\(^2\).

\(^1\)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.

\(^2\)Also known as strategy or policy.
An example are all paths that end in a particular terminal state \( s_T \) within \( \tau \) steps.

C. Control Improvisation

In control improvisation, we aim to find an ego-policy, \( \sigma_{\text{ego}} \), 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 \( X_{1:i} \equiv X_1, \ldots, X_i \) and \( Y_{1:i} \equiv Y_1, \ldots, Y_i \) denote two sequences of random variables. The probability of \( X_{1:i} \) causally conditioned on \( Y_{1:i} \) is:

\[
\Pr(X_{1:i} \mid Y_{1:i}) \equiv \prod_{j=1}^{i} \Pr(X_j \mid X_{1:j-1} Y_{1:j}).
\]

The causal entropy of \( X_{1:i} \) given \( Y_{1:i} \) is then defined as,

\[
H(X_{1:i} \mid Y_{1:i}) \equiv \mathbb{E}_{X_{1:i}, Y_{1:i}}[-\log(\Pr(X_{1:i} \mid Y_{1:i}))]
\]

Using the chain rule, one can relate causal entropy to (non-causal) entropy, \( H(X_{1:i} \mid Y_{1:i}) \), via:

\[
H(X_{1:i} \mid Y_{1:i}) = \sum_{t=1}^{i} H(X_t \mid Y_{1:t-1}, X_{1:t-1})
\]

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,

\[
H(X_{1:i} \mid Y_{1:i} \mid Z) \equiv \sum_{t=1}^{i} H(X_t \mid Y_{1:t-1}, X_{1:t-1}, Z).
\]

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 \( G \) and a scheduler \( \sigma \), let us denote by \( A_{1:i}^{\text{ego}} \) and \( S_{1:i} \) random variable sequences for ego-player actions and states respectively. The causal entropy of controllable actions in \( \tau \)-length paths under \( \sigma \) is then

\[
H_{\tau}(\sigma) \equiv H(A_{1:\tau}^{\text{ego}} \mid S_{1:\tau}),
\]

where \( \tau' = \lfloor \frac{\tau}{2} \rfloor \) is the number of ego-actions due to alternation.

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

We now formalize the problem statement.

3Such 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 \( G \), \( \tau \)-bounded path properties \( \psi \) and \( \varphi \), and thresholds \( p \in [0, 1] \) and \( h \in [0, \infty) \), find a ego-policy \( \sigma_{\text{ego}} \) (or report that none exists) such that for every env-policy \( \sigma_{\text{env}} \),

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

where \( \sigma = (\sigma_{\text{ego}}, \sigma_{\text{env}}) \).

We say that an instance of the ERCI problem is realizable, if an appropriate \( \sigma_{\text{ego}} \) exists and call such \( \sigma_{\text{ego}} \) 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 \( \tau \)-bounded path properties (monitorable by finite automata), this assumption is naturally realized by a \( \tau \)-step unrolling of a monitor augmented SG [4] i.e., augmenting the state space with a counter from 0 to \( \tau \) and the current property monitor state.

Next, in order to ensure the hard constraint, \( \psi \), 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 \( G \) 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 \( s_T \) and \( s_\perp \), based on membership in \( \varphi \), i.e.,

\[
\text{last}(\xi) = s_T \implies \xi \in \varphi
\]

\[
\text{last}(\xi) = s_\perp \implies \xi \not\in \varphi.
\]

Example 5. In Fig. 3a we show a (deterministic) MDP and we plot for all schedulers the induced probability to reach \( s_T \) 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 \( s_T \), 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 \( \varphi \) (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 \( p \) and \( h \) that yield a

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].
While the performance and randomization may be better than
We observe that guaranteed points are downward closed, i.e.,
we cast ERCI as an instance of a multi-objective optimization
Example To ease notation, for
Proposition 1. The set of achievable points, \( S \), is convex.
Next, because \( S \) 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 \( x < x' \), i.e., if \( x \leq x' \) and \( x \neq x' \).
The Pareto front \( F_S \) of \( S \) is then the set of non-dominated
achievable points,
\[
F_S \overset{\text{def}}{=} \{ x \in S \mid \forall x' \in S, x \not< x' \}.
\] (11)
Importantly, it holds that the ERCI problem is satisfiable
iff there exists a \( x \in F_S \) such that \( \langle p, h \rangle \leq x \).

Example 7. The set \( S \) 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
\( \langle p, h \rangle \) in the green area is realizable.
Approximating the Pareto front gives a natural approximation
scheme for ERCI instances: For any subset \( F \subseteq F_S \),
1) If there exists an \( x \in F \) such that \( \langle p, h \rangle \leq x \), then
the ERCI problem must be realizable and \( x \) is a witness
to realizability.
2) If there exists an \( x \in F \) such that \( x < \langle p, h \rangle \), 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 \( x_1, x_2 \in F \) such that \( \langle p, h \rangle \leq x \times (1 - w) \cdot x_2 \), 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 \( \sigma_{\lambda_1} \) and \( \sigma_{\lambda_2} \) such that \( \langle p, h \rangle \leq w \times x_1 + (1 - w) \times x_2 \), then the policy described by
\[
\sigma_{\text{ego}}(\alpha | s) \overset{\text{def}}{=} w \cdot \sigma_{\lambda_1}(\alpha | s) + (1 - w) \cdot \sigma_{\lambda_2}(\alpha | s)
\] (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 \( S \): It exploits the downward
closure and the convexity of \( S \). 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 \( w = \langle w_1, w_2 \rangle \),

Fig. 3. Minimal ERCI problem with \( \varphi = (\text{last}(\xi) = s_{\uparrow}) \)
We remark that the reparameterization is not only beneficial where $h$ as we have seen, these restrict the domain in which we obtain the following (core) ERCI problem.

$\epsilon$ the point of the white point is given by Geometrically, after computing from a usability point-of-view, but it also eases our exposition. Following

and intuitively, we can trade between these two points following $\langle p, h \rangle$ and $\langle p^*, h^* \rangle$, respectively. Likewise, we define $S$ is a finite polytope – the Pareto front.

Thus a key algorithmic question in ERCI is how to efficiently explore the Pareto front $F_S$.

### C. Regret-Based ERCI

To algorithmically explore the Pareto-curve, we reparameterize 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 $h = \max \{h \mid \exists p \text{ s.t. } \langle p, h \rangle \in S\}$, i.e., the largest randomness that can be guaranteed by any ego-policy. Likewise, we define $p^* = \max \{p \mid \exists h \text{ s.t. } \langle p, h \rangle \in S\}$, i.e., the largest performance that can be guaranteed by any ego-policy. Then, we define $p^- = \max \{p \mid \langle p, h^* \rangle \in S\}$, the best performance that ego can guarantee while guaranteeing optimal randomness. Likewise, we define the analogous $h^- = \max \{h \mid \langle p^*, h \rangle \in S\}$. 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 $p$ and $h$ a priori, we seek to guarantee some percentage of the independently achievable soft constraint and causal entropy measure. We re-interpret ERCI as follows:

$$p_\epsilon = \epsilon \cdot (p^* - p^-) + p^- \quad h_\delta = \delta \cdot (h^* - h^-) + h^- \quad (13)$$

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. 4 is definitely realizable, and the regret-based ERCI asks whether the white circle is also realizable (where the point of the white point is given by $\epsilon$ and $\delta$. Together, we obtain the following (core) ERCI problem.

The Core ERCI Problem: Given an finite acyclic SG $\mathcal{G}$, with terminal states, $s_T$ and $s_\bot$, and thresholds $\epsilon, \delta \in [0, 1]$, find an ego-policy $\sigma_{ego}$ s.t. for every env-policy $\sigma_{env}$:

1) (soft constraint) $\Pr(\text{last}(\xi) = s_T \mid \sigma) \geq p_\epsilon$

2) (randomness constraint) $H(\sigma) \geq h_\delta$

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

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, $\sigma$, yields the optimal randomness ratio, $\delta$:

$$f_\delta : [0, 1] \to [0, 1]$$

$$f_\delta(\delta) = \max_{\epsilon} \{h_\delta \mid \langle p_\epsilon, h_\delta \rangle \in S\} \quad (14)$$

**Proposition 2.** $f_\delta$ 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 $S$ is (in general) not a finite polytope – the MDP in Fig. 3a serves as an example. Nevertheless, $S$ can be well approximated with finitely many vertices, see Ex. 3.

With these facts, we are now well-equipped to develop the algorithms in Sec. 5 for MDPs and Sec. 6 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 $F_S$ and incrementally build up $\mathcal{F} \subseteq F_S$.

### 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 $\lambda$ of the policy. Formally, the rationality corresponds to the following scalarization of our multi-objective problem [37],

$$J_\lambda(\sigma) \overset{\text{def}}{=} \langle 1, \lambda \rangle \cdot \langle h_\sigma, p_\sigma \rangle. \quad (15)$$
In context of MDPs, the unique (ego-)policy that optimizes \([15]\) is given by a smooth variant of the Bellman equations \([60]\). Namely, let \(\text{smooth}\) denote the log-sum-exp operator, i.e., 
\[
\text{smooth}(X) \overset{\text{def}}{=} \log \left( \sum_{x \in X} e^x \right)
\]
For each rationality \(\lambda \in [0, \infty)\), we define a policy \(\sigma_{\lambda}\) as follows:
\[
\sigma_{\lambda}(s | a) \overset{\text{def}}{=} \exp(Q_{\lambda}(s, a) - V_{\lambda}(s))
\]
\[
V_{\lambda}(s) \overset{\text{def}}{=} \begin{cases} 
\lambda \cdot \{s = s_T\} & \text{if } s \in \{s_T, s_{\pm}\}, \\
\text{smooth}_{a \in A(s)} Q_{\lambda}(s, a) & \text{otherwise.}
\end{cases}
\]
\[
Q_{\lambda}(s, a) \overset{\text{def}}{=} \sum_{s'} P(s, a, s') \cdot V_{\lambda}(s').
\]
To ease notation, we denote \(x_{\lambda} \overset{\text{def}}{=} x_{\sigma_{\lambda}}, p_{\lambda} \overset{\text{def}}{=} p_{\sigma_{\lambda}}, h_{\lambda} \overset{\text{def}}{=} h_{\sigma_{\lambda}}\). Intuitively, as \(\lambda \to 0\), \(\sigma_{\lambda}\) approaches the uniform distribution over all available actions. Note that this policy maximizes (causal) entropy, and thus \(h^{\ast} = h_0\). As \(\lambda \to \infty\), this variant of the Bellman equations coincides with the standard Bellman equations \([47]\), where \(\sigma_{\lambda}\) selects (uniformly) from actions that maximize performance. Furthermore, the monotonicity and smoothness of the above Bellman equations yields the following proposition.

**Proposition 3.** \(p_{\lambda}\) is continuously (and strictly) increasing in \(\lambda\) and \(h_{\lambda}\) is smoothly (and strictly) decreasing in \(\lambda\).

In terms of \(f_{\S}\), we can define:
\[
\epsilon_{\lambda} \overset{\text{def}}{=} \left( \frac{p_{\lambda} - p_0}{p_{\infty}} + p_0 \right) \quad \text{and} \quad \delta_{\lambda} \overset{\text{def}}{=} \frac{h_{\lambda} - h_{\infty}}{h_0} + h_{\infty}.
\]

Then, because \(\sigma_{\lambda}\) maximizes randomness given a target performance, one derives:
\[
f_{\S}(\delta_{\lambda}) = \epsilon_{\lambda}.
\]

What remains is to instantiate the approximation scheme for the Pareto front by varying the optimization direction \((\lambda, 1)\)\(^6\). In particular, we construct \(\mathcal{F} = \{x_{\lambda} | \lambda \in \{\lambda_1, \lambda_2, \ldots\}\}\) 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 \(\lambda\).

### B. Targeted Pareto-exploration

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

We propose a three staged sequence: (i) Compute \(x_{\lambda}\) for the end points \(\lambda \in \{0, \infty\}\). (ii) Double \(\lambda\) (starting at \(\lambda = 1\)) until \(h_{\lambda} \leq h\), 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 \((p, h)\) is not on \(\mathcal{F}_{\S}\) (or if we happen to exactly hit \((p, h)\) by selecting some rationality \(\lambda\)). 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 \(p^\ast, h^\ast \neq 0\) (which would otherwise yield trivial \(\S\) and \(\mathcal{F}_{\S}\))

![Fig. 5. SG to illustrate entropy matching policies.](image)

Our approximation scheme yields a semi-decision process which halts either \((a) (p, h)\) is bounded away from \(\mathcal{F}_{\S}\) or \((b) (p, h)\) is dominated by \(x_{\lambda}\).

Next, observe that if we terminate the binary search when the search region is smaller than \(\Delta\), this approximation scheme becomes linear in the MDP size and logarithmic in the final rationality, \(\lambda^\ast\), and the resolution, \(\Delta\), i.e., the run-time is,
\[
O \left( \log(\mathcal{G}) \cdot \log(\lambda^\ast) \cdot \log(1/\Delta) \right)
\]

Finally, before generalizing to stochastic games, we observe that in practice, \(\lambda = 100\) yields a nearly optimal policy, and thus one can often assume \(\lambda^\ast \leq 100\) 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 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 requirement. Second, if there is a violating env-policy, there is a deterministic env-policy that proves this. In particular, at every state, \(\sigma_{\text{env}}\) 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 \(\mathcal{G}[\sigma_{\text{env}}]\).

**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, \(\sigma_{\text{env}}\), that lexicographically minimizes the guaranteed randomness, followed by performance. On the sub-graph, \(\mathcal{G}[\sigma_{\text{env}}]\), ego employs the corresponding entropy maximizing policy for the MDP \(\mathcal{G}[\sigma_{\text{env}}]\). 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 $\lambda$, 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 $\lambda$, unless state $s_4$ is reached. As this ensures a higher entropy, we may now select a higher rationality, $\lambda'$.

**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 $f_\delta(h_\lambda)$ 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 subgraphs. 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 $\lambda$, 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 $\kappa$ error along the performance axis.

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

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

where $z = q \cdot x + \bar{q} \cdot y$. Thus, so long as $\kappa \cdot \tau$ is enough resolution to answer $p_\lambda < p$, 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 $\kappa^*$ tolerance is required to terminate, then the $\kappa$ search introduces $O(\log(1/\kappa^*))$ 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:

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

where, $N_\lambda$ is the number of unique rationality coefficients processed. If, as in the MDP case, one assumes a maximum rationality coefficient $\lambda^*$ and a minimum rationality resolution $\Delta$, one obtains:

$$O\left(\log(1/\kappa^*) \cdot \lambda^*/\Delta \cdot \log(\lambda^*/\Delta) \cdot |\mathcal{G}|\right).$$

The above however is very conservative and empirically we observe $N_\lambda$ bounded far away from $\lambda^*/\Delta$.

**VII. Implementation and Empirical Evaluation**

**Setup.** To experimentally validate the feasibility of our ERCI algorithm for SGs, we implemented [55] our algorithm in Python. Inspired by the recent work on compressing MDPs
for specification inference \cite{57}, each SG was represented as a Binary Decision Diagram (BDD) \cite{9} using the dd and py-aiger python packages \cite{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 $\{[k/4, [k/4]\}'$, and the drones $D_{\text{ext}}$ and $D_{\text{env}}$ 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.\footnote{Note that counter-intuitively, only using soft constraints generally results in harder instances as the compressed SGs are larger.} 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 demonstra\textsuperscript{8}tes, 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).\footnote{For example, the ($k = 8$, horizon = 18) case is encoded using a 505,100 node BDD ($|G| = 6,861$ nodes). Compare with the direct encoding $|G| = |S| \cdot \tau \cdot |\text{monitor state}| = (8 \times 8)^2 \cdot (2 \cdot 18) \cdot (2^4 \cdot 2) \approx 37,000$.}

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 \cite{22} and imitating power usage in households \cite{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 \cite{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 $G$, with terminal states, $s_T$ and $s_L$, and thresholds $p \in (0, 1)$ and $h \in [0, \infty)$, find a ego-policy $\sigma_{\text{ego}}$ such that for every env-policy $\sigma_{\text{env}}$

1) (soft constraint) $\Pr(\text{last}(\xi) = s_T | \sigma) \geq p$.

2) (randomness) $\max_{\xi} \Pr(\xi | \sigma) \leq d$,

where $\sigma = (\sigma_{\text{ego}}, \sigma_{\text{env}})$.\footnote{However, this does not mean that the algorithm to compute a solution carries over to the general case}

While RCI is only applied to deterministic SGs in \cite{19}, there is nothing in the definition that prevents its application to the general class of SGs.\footnote{We observe that then, the only difference is}
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 $d \geq 1/n$. 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 $\delta = 1/n$ satisfies it for any $d \geq 1/n$. 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,

$$f: (d, G) \mapsto h,$$

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

### 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 [49], cost-bounded reachability [26], or mean-payoff properties [1]. 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 [56]. 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 guarantees, but are more search-based.

Another related domain is the problem of (randomly) patrolling a perimeter and points of interest [11, 5]. 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 [8], 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 [33]. 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 variant of this literature served as the basis for our MDPs.
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 extending the theory to games with imperfect information. 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.

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.

REFERENCES

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


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


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


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. Consider two points \((p, h), (p', h') \in S\) achieved by \(\sigma_{ego}\) and \(\sigma'_{ego}\) respectively. Consider the new policy, \(\pi\), defined by employing \(\sigma_{ego}\) with probability \(q\) and \(\sigma'_{ego}\) with probability \(\bar{q} \equiv 1 - q\). Because each policy guarantees its corresponding performance, this new policy has performance at least \(q \cdot p + \bar{q} \cdot p'\). Similarly, by viewing \(\pi\) as a random variable and applying chain rule yields,

\[
H_\tau(\sigma) \geq q \cdot H(A_{1:\tau}^{ego}) \mid S_{1:\tau} \mid \pi = \sigma_{ego} + \\
\bar{q} \cdot H(A_{1:\tau}^{ego}) \mid S_{1:\tau} \mid \pi = \sigma'_{ego} \\
= q \cdot h + \bar{q} \cdot \bar{h}'.
\]

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 \(G\). To simplify our proof, observe that w.l.o.g., we can restrict our attention to ERCI instances on the Pareto front, \((p, h) \in F_3\). Next, for the sake of contradiction, we shall assume that no entropy matching policy achieves \((p, h)\), but \(\sigma_{ego}^*\) does:

\[
\forall \sigma_{ego} \in \{\sigma_{ego}^{\lambda}\} \cdot x_{\sigma_{ego}} \prec (p, h) \tag{26}
\]

\[
\exists \sigma_{ego}^* \notin \{\sigma_{ego}^{\lambda}\} \cdot (p, h) \preceq x_{\sigma_{ego}^*}. \tag{27}
\]

Indeed, we may reformulate (27) to

\[
\exists \sigma_{ego}^* \notin \{\sigma_{ego}^{\lambda}\} \cdot (p, h) = x_{\sigma_{ego}^*} \tag{28}
\]

as we assumed that \((p, h)\) 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, \(\lambda\), such that \(\sigma_{ego}^*\) induces entropy \(h\):

\[
h_{\sigma_{ego}^*} = h = h_{\sigma_{ego}^*}, \tag{29}
\]

where the second equality follows from (28). Next, let \(\sigma_{env}^{\lambda}\) denote the min-entropy env-policy given \(\sigma_{ego}^*\), i.e., the policy that minimizes entropy in \(G[\sigma_{ego}^*]\). Because \(\sigma_{ego}^*\) witnesses \((p, h)\), it must be the case that:

\[
h_{(\sigma_{ego}^*, \sigma_{env}^{\lambda})} \geq h \quad \text{and} \quad p(\sigma_{ego}^*, \sigma_{env}^{\lambda}) \geq p \tag{30}
\]

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:

\[
h = h_{(\sigma_{ego}^*, \sigma_{env}^{\lambda})} \geq h_{(\sigma_{ego}^{\lambda}, \sigma_{env}^{\lambda})} \geq h, \tag{31}
\]

and thus,

\[
h_{(\sigma_{ego}^*, \sigma_{env}^{\lambda})} = h_{(\sigma_{ego}^{\lambda}, \sigma_{env}^{\lambda})}. \tag{32}
\]

Thus, from uniqueness on MDPs, \(\sigma_{ego}^*\) and \(\sigma_{ego}^{\lambda}\) must exactly match on \(G[\sigma_{env}^{\lambda}]\) 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 \(\sigma_{ego}^*\) achieves a given \((p, h)\) on this subgraph, there must be an entropy matching that does so as well. Thus,

\[
x_{\sigma_{ego}^*} \preceq x_{\sigma_{ego}^{\lambda}}, \tag{33}
\]

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