Let's Collaborate: Regret-based Reactive Synthesis for Robotic Manipulation

Karan Muvvala, Peter Amorese, and Morteza Lahijanian

***Abstract*— As robots gain capabilities to enter our human-centric world, they require formalism and algorithms that enable smart and efficient interactions. This is challenging, especially for robotic manipulators with complex tasks that may require collaboration with humans. Prior works approach this problem through reactive synthesis and generate strategies for the robot that guarantee task completion by assuming an adversarial human. While this assumption gives a sound solution, it leads to an "unfriendly" robot that is agnostic to the human intentions. We relax this assumption by formulating the problem using the notion of *regret*. We identify an appropriate definition for regret and develop regret-minimizing synthesis framework that enables the robot to seek cooperation when possible while preserving task completion guarantees. We illustrate the efficacy of our framework via various case studies.**

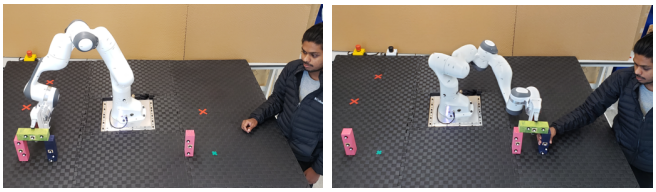
I. Introduction

From factories to households, robots are rapidly leaving behind their robot-centric environments and entering our society. Examples include self-driving cars, delivery robots, and assistive robots. To be successful in our human-centric world, robots must develop the ability to interact with dynamic environments. This includes performing complex tasks in the presence of humans, who have their own objectives and may interfere with the robot's task. Therefore, robots must aim to have effective interactions with humans and seek collaboration whenever possible. To achieve such capabilities, strategies that account for human objectives as well as task and resource constraints are needed. Generating such strategies, however, is challenging due to two main reasons: *formulation* and *computation*. That is, a proper mathematical formulation of such strategies is a nontrivial problem, and computation cost for strategies that enable reactivity is inherently high, especially in the manipulation domain, where tasks are complex and space of reasoning is high dimensional. This work mainly focuses on the formulation challenge and also aims to design a reactive synthesis framework that enables robots to seek collaboration with humans while guaranteeing completion of their task.

As an example, consider the scenario in Fig. 1, where the robot is tasked with building an arch either on the left or right side of the table with a green block on top. In this workspace, a human can reach and manipulate the blocks placed on the right side but not the ones on the left. To operate in the left side, the robot has to spend more energy than the one closer to the human. For such scenarios, classical planning methods that compute a fixed sequence of actions are not sufficient. Instead, we need the robot to reason and react to the human's actions for efficient interactions.

This work was supported in part by University of Colorado Boulder and NASA COLDTech Program under grant #80NSSC21K1031.

Authors are with the department of Aerospace Engineering Sciences at the University of Colorado Boulder, CO, USA {firstname.lastname}@colorado.edu



(a) Adversarial Behavior (b) Probabilistic Behavior

Fig. 1: Arch Construction. (a) Strategy via [2] (human assumed to be adversarial). (b) Policy via [8] (robot expects human to be cooperative based on prior data).

Previous work [1]–[3] addresses this problem through the lens of *reactive synthesis* [4]–[6]. They employ *Linear Temporal Logic over finite traces* (LTLf) [7] to specify tasks that can be accomplished in finite time. The framework models the interaction between the human and the robot as a two-player game. By assuming the human to be purely adversarial in this game, a strategy is synthesized for the robot that guarantees task completion under all possible human moves. This assumption however is conservative, eliminates any room for collaboration, and results in "unfriendly" behaviors that could lead to higher energy spending than allowing a chance for collaboration. In Fig. 1a, using that approach, the robot builds the arch away from the human irrespective of the human's intention.

Recent work [8] relaxes this assumption by modelling the human as a probabilistic agent. This leads to an abstraction in the form of a Markov Decision Process (MDP), and the objective reduces to synthesizing an optimal policy that maximizes the probability of satisfying the task. The approach optimizes the robot's actions according to the expected behavior of the human instead of assuming they are always adversarial. In the scenario in Fig. 1, using this approach, the robot builds the arch near the human (Fig. 1b) if the expected behavior of the human is to be cooperative; otherwise, the robot builds the arch away from the human. While efficient interactions can be achieved using that framework, the required prior knowledge on the human is generally hard to obtain. The method also fails to capture the human as a strategic agent with their own objectives.

In the machine learning and game theoretic communities, an emerging method of reasoning about the quality of strategies (actions) is via the notion of regret [9]–[13]. *Regret* is the measure of "goodness" of an action in comparison to the best response that could have been received in hindsight

[9], [14]. This is different from the classical notion of cost or reward. In regret games, instead of trying to minimize the total cost, the objective is to minimize regret. The obtained strategies have shown to be more reasonable than, e.g., Nash Equilibrium strategies, for games played for finite number of cycles [15]. Various formalisms for regret have been introduced in different communities. The reinforcement learning community specifically uses a formalism that is suitable for exploitation of the degree of incomplete information in games in regards to partial observability or uncertainty in transition probabilities [11], [16]. Those approaches look at regret locally, whereas the formal methods community views regret globally through the lens of reactive systems [17], [18]. Nevertheless, the notion of regret has not been studied for robotic manipulation, especially in the context of reactive synthesis for tasks with human interventions.

In this work, we propose a regret-based reactive synthesis framework that enables a robotic manipulator to seek collaboration with the human while guaranteeing task completion and staying within its resource limits. This framework relaxes the assumption that the human is purely adversarial while still capturing the human as a strategic agent without requiring *a priori* knowledge. As in [1]–[3], [8], we consider tasks given as LTLf formulas. Our approach is based on abstracting the interaction between the robot and human as a two-player game. We then formulate a regret game on the abstraction by defining an appropriate formalism for regret in the context of interactive manipulation domain. We adapt an algorithmic approach to generate regret-minimizing strategies if they exist. We show that these strategies guarantee task completion and enable efficient interactions, but they are history dependent, which means they are computationally expensive. Finally, we illustrate the benefits of regret-minimizing strategies in several case studies and compare the results against adversarial strategies.

The contributions of this work are threefold. First, we identify an appropriate regret formalism for the manipulation domain, and based on that, we introduce a regret-minimizing reactive synthesis framework that encourages collaboration while still guaranteeing the satisfaction of the task and resource requirements. This framework paves the way for further studies on using regret in robotics since regret-minimizing behaviors are natural and human-like [15]. Second, we provide an end-to-end toolbox for synthesizing regret-minimizing strategies. Finally, we illustrate the regret-minimizing strategies and their corresponding emergent behaviors through several case studies [19], [20].

II. Problem Formulation

The goal of this work is to synthesize a high-level strategy for a robotic manipulator to achieve a task defined over a set of objects via an efficient interaction with a human. We assume that the human has their own objective that may not be necessarily adversarial to the robot's task. Hence, we want to enable the robot to explore possible collaboration with the human in the execution of the task while guaranteeing task completion. Below, we formalize this problem by first introducing the required mathematical definitions.

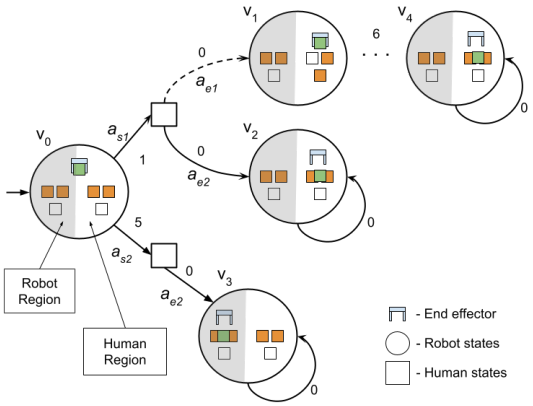


Fig. 2: Example illustration of abstraction $\mathcal{G}$ for the Arch construction example in Fig. 1. The human can move objects (colored blocks) in its region (unshaded) but cannot reach the objects placed in robot (shaded) region. Robot actions a_{s1} and a_{s2} denote placing the green block in the human region and robot region, respectively. The edge weights are the transition (action) costs.

A. Manipulation Domain Abstraction

The manipulation domain describes how the robot, human, and objects can interact with each other. This domain is naturally continuous, and previous works [2], [3] show how a discrete abstraction of it can be constructed. The abstraction captures the interaction as a turn-based two-player game between the robot and human at discrete time steps.

Definition 1 (Manipulation Domain Abstraction)**.** *The manipulation domain abstraction is a tuple* $\mathcal{G} = (V, v_0, A_s, A_e, \delta, F, \Pi, L)$ *where:*

- $V = V_s \cup V_e$ *is the set of states partitioned into robot (system) states* V_s *and human (environment) states* V_e*,*
- A_s *and* A_e *are the sets of finite actions for the robot and human, respectively,*
- $\delta : V \times (A_s \cup A_e) \to V$ *is the transition function,*
- $F : V \times (A_s \cup A_e) \times V \to \mathbb{R}_{\geq 0}$ *is the cost function that maps each transition to a cost. Here, the cost of transitions enabled by the human actions are assumed to be zero, i.e.,* $F(v, a_e, v') = 0 \ \ \forall a_e \in A_e$ *,*
- Π *is a set of task related atomic propositions, and*
- $L : V \to 2^{\Pi}$ *is a labeling function that indicates the property of each state relative to the task.*

Each state of $\mathcal{G}$ represents a configuration of the world, i.e., object locations and robot end-effector, and each transition corresponds to an action taken by the robot agent or the human agent. A transition from one state to another represents evolution of the shared workspace under that action.

Example 1. *Fig. 2 shows a simple abstraction* $\mathcal{G}$ *for the scenario in Fig. 1. The robot starts from state* v_0 *with the green block in its end-effector. The robot has two choices from* v_0*: place the block near the human (unshaded region)*

or away from the human (shaded region). Once the robot takes an action, it is the human's turn to decide whether to intervene or not. The solid and dashed edges indicate that the human does not intervene and intervenes, respectively.

We note that this turn-based game abstraction is able to capture multiple consecutive human moves as well as concurrent actions by the agents in the continuous domain with the assumption that human actions are faster than robot actions as in [2]. Prior work [3] shows an automated process for constructing this abstraction and representing it in the Planning Domain Definition Language (PDDL) [21].

B. Strategy and Payoff Function

We assume both the human and robot are strategic agents, i.e., they choose actions in accordance with their own objectives. Informally, a strategy is a mapping that chooses a valid action to perform given the history of executions so far. A finite execution of the game is a finite sequence of states of $\mathcal{G}$. The set of all finite executions is denoted by V^*.

Definition 2 (Strategy)**.** *A strategy for the robot is a function $\sigma : V^* \cdot V_s \to A_s$ that maps a finite execution ending in a robot state in V_s to a robot action in A_s. Similarly, a strategy for the human is a function $\tau : V^* \cdot V_e \to A_e$ that maps a finite execution ending in a human state in V_e to a human action in A_e. A strategy is said to be memoryless if it only depends on the current state. Otherwise, it is called a finite memory strategy. We denote by $\Sigma^{\mathcal{G}}$ and $\mathrm{T}^{\mathcal{G}}$ the set of all strategies for the robot and the human, respectively, in $\mathcal{G}$.*

The realization of the strategies σ and τ on abstraction $\mathcal{G}$ is called a play, which is a sequence of states that correspond to the evolution of the world under the actions executed by the robot and human as per their strategies.

Definition 3 (Play)**.** *A play (a.k.a run or trajectory) on $\mathcal{G}$ is an infinite sequence of states $r(\sigma, \tau) = v_0 v_1 v_2 \ldots \in V^\omega$ induced by strategies σ and τ such that it is consistent with the strategies, i.e., for all $i \geq 0$, $v_{i+1} = \delta(v_i, \sigma(v_0 \ldots v_i))$ if $v_i \in V_s$, otherwise $v_{i+1} = \delta(v_i, \tau(v_0 \ldots v_i))$.*

For a play $r(\sigma, \tau) = v_0 v_1 \ldots$, we call the obtained sequence of observations $\rho(r(\sigma, \tau)) = L(v_0)L(v_1)\ldots$, where each $L(v_i) \in 2^{\Pi}$, the *trace* of r. We now define a payoff function that helps quantitatively reason over different plays in $\mathcal{G}$ induced by various strategies. Informally, the payoff function is the total energy cost that the robot incurs in a play.

Definition 4 (Payoff)**.** *Given robot and human strategies σ and τ, the payoff function at state v_0 is the total cost of the robot actions in the induced play $r(\sigma, \tau) = v_0 v_1 \ldots$, i.e,*

$$\mathrm{Val}^{v_0}(\sigma, \tau) = \sum_{i=1}^{\infty} F(v_{i-1}, a_i, v_i), \tag{1}$$

where $a_i = \sigma(v_0 \ldots v_{i-1})$ if $v_{i-1} \in V_s$, otherwise $a_i = \tau(v_0 \ldots v_{i-1})$.

Example 2. *For abstraction $\mathcal{G}$ in Fig. 2, a memoryless strategy τ for the human is to always intervene (a_{e1}) while a finite-memory strategy τ is to not intervene (a_{e2}) if the strategy σ for the robot is to build the arch near the human. The payoff* Val *associated with the strategy that the robot always picks action a_{s2} and human picks action a_{e2} is 5.*

C. Manipulation Task

We consider manipulation tasks that can be achieved in finite time. To express such tasks, we use LTLf [7], which is a language that combines Boolean connectives with temporal operators, allowing expression of complex tasks.

Definition 5 (LTLf Syntax [7])**.** *The syntax of LTLf formula φ is defined recursively as:*

$$\varphi := p \mid \neg\varphi \mid \varphi \ \wedge \ \varphi \mid X \ \varphi \ \mid \varphi U \varphi \tag{2}$$

where $p \in \Pi$ is an atomic proposition, "$\neg$" (negation) and "$\wedge$" (and) are Boolean operators, and "X" (next) and "U" (until) are the temporal operators.

The commonly used temporal operators "F" (eventually) and "G" (globally) are defined as: $F\varphi := \top \ U \ \varphi$ and $G\varphi := \neg F \neg \varphi$. The LTLf semantics is as follows.

Definition 6 (LTLf Semantics [7])**.** *LTLf formulas are defined over finite sequence of observations called a trace $\rho \in (2^{\Pi})^*$. Let $|\rho|$ denote the length of the trace ρ, $\rho[i]$ be the i^{th} observation in ρ, and $\rho, i \models \varphi$ denote that the i^{th} step of trace ρ that satisfies the formula φ. Then,*

- $\rho, i \models \top$,
- $\rho, i \models p$ *iff* $p \in \rho[i]$,
- $\rho, i \models \neg\varphi$ *iff* $\rho, i \not\models \varphi$,
- $\rho, i \models \varphi_1 \wedge \varphi_2$ *iff* $\rho, i \models \varphi_1$ *and* $\rho, i \models \varphi_2$,
- $\rho, i \models X\varphi$ *iff* $|\rho| > i+1$ *and* $\rho, i+1 \models \varphi$,
- $\rho, i \models \varphi_1 \ U \ \varphi_2$ *iff* $\exists \ j$ *s.t.* $i \leq j < |\rho|$, *and* $\rho, i \models \varphi_2$ *and* $\forall k, \ i \leq k < j, \rho, k \models \varphi_1$.

The set of finite traces that satisfy φ is called the language of φ, i.e., $\mathcal{L}(\varphi) = \{\rho \in (2^{\Pi})^* \mid \rho \vDash \varphi\}$.

We say that a play $r(\sigma, \tau)$ of $\mathcal{G}$ satisfies LTLf formula φ iff there exists a prefix of its trace $\rho(r(\sigma, \tau))$ that satisfies φ, i.e., $r(\sigma, \tau) \models \varphi$ iff

$$\exists \, pre(\rho(r(\sigma, \tau))) \ \text{ s.t. } \ pre(\rho(r(\sigma, \tau))) \models \varphi,$$

where $pre(\rho(r(\sigma, \tau)))$ is a prefix of trace $\rho(r(\sigma, \tau))$.

Example 3. *For the arch-building example in Fig. 1, the task can be written as the LTLf formula*

$$\begin{aligned}\varphi_{arch} = & F\left(p_{\text{green, top}} \wedge p_{\text{block, support}_1} \wedge p_{\text{block, support}_2}\right) \wedge \\ & G\left(\neg(p_{\text{block, support}_1} \wedge p_{\text{block, support}_2}) \rightarrow \neg p_{\text{green, top}}\right).\end{aligned}$$

D. Problem

We are interested in generating robot strategies that encourage collaboration with humans. At the same time, we require the robot to complete the task while never exceeding a given energy budget. We consider the following problem.

Problem 1. *Given abstraction $\mathcal{G}$, LTLf task specification φ, and a user-defined energy budget $\mathcal{B}$, compute a strategy σ for*

the robot that not only guarantees completion of task φ but also explores possible collaborations *with the human while keeping the total energy* $\mathrm{Val}^{v_0}(\sigma,\tau) \leq \mathcal{B}$ *for all* $\tau \in \mathrm{T}^{\mathcal{G}}$.

III. Regret-based Reactive Synthesis

In this section, we introduce our solution to Problem 1, by building on previous work [2] and formulating a regret game. We discuss the intuition for the regret formulation to produce cooperation-seeking behaviors for the robot. Our proposed approach first converts the LTLf formula φ to a discrete structure that graphically represents the task, and then composes it with with abstraction $\mathcal{G}$. On the resulting structure, we formulate a regret game with a proper definition for regret that incorporates the desired attribute of seeking cooperation while guaranteeing task completion.

A. DFA Game

Every LTLf formula φ can be converted to a *Deterministic Finite Automaton* (DFA) that captures all the possible ways in which one can satisfy φ [7]. Given task φ, we construct DFA $\mathcal{A}_\varphi = (Z, z_0, \Sigma, \delta, Z_f)$, where Z is a finite set of states, z_0 is the initial state, $\Sigma = 2^{\Pi}$ is the alphabet, $\delta : Z \times \Sigma \to Z$ is the deterministic transition function, and $Z_f \subseteq Z$ is the set of accepting states. A run of $\mathcal{A}_\varphi$ on a trace $\rho = \rho[1]\rho[2]...\rho[n]$, where $\rho[i] \in 2^{\Pi}$, is a sequence of DFA states $z_0 z_1 \dots z_n$, where $z_{i+1} = \delta(z_i, \rho[i])$. If $z_n \in Z_f$, then trace ρ satisfies φ, i.e., $\rho \models \varphi$.

Given a DFA $\mathcal{A}_\varphi$ and the manipulation domain abstraction $\mathcal{G}$, we compose the two graphs to construct the DFA Game $\mathcal{P}$ that captures all the possible ways in which task φ can be accomplished on $\mathcal{G}$. The states in $\mathcal{P}$ represent the current configuration of the physical world as well as how much of the task φ has been accomplished so far.

Definition 7 (DFA Game)**.** *A DFA Game is a tuple* $\mathcal{P} = \mathcal{G} \times \mathcal{A}_\varphi = (S, S_f, s_0, A_s, A_e, F_\mathcal{P}, \delta_\mathcal{P})$ *where* A_s *and* A_e *are as in Def. 1, and*

- $S = V \times Z$ *is a finite set of states, and* $S_s = V_s \times Z$ *and* $S_e = V_e \times Z$ *are the set of robot and human states, respectively,*
- $s_0 = (v_0, z_0)$ *is the initial state,*
- $S_f = \{(v, z) \,|\, z \in Z_f\}$ *is the set of final or accepting states,*
- $\delta_\mathcal{P} : S \times (A_s \cup A_e) \to S$ *is the deterministic transition function such that a transition from* $s = (v, z)$ *to* $s' = (v', z')$ *under* $a \in (A_s \cup A_e)$ *exists if* $v' = \delta(v, a)$ *and* $z' = \delta(z, L(v))$ *and* $z \notin Z_f$*. If* $z' \in Z_f$*, then* $\delta_\mathcal{P}((v, z), a) = (v, z)$ *for all* $a \in (A_s \cup A_e)$*,*
- $F_\mathcal{P} : S \times (A_s \cup A_e) \times S \to \mathbb{R}_{\geq 0}$ *is the cost function such that* $F_\mathcal{P}((v, z), a, (v', z')) = F(v, a, v')$.

A run on $\mathcal{P}$ is a valid sequence of states obtained with $\delta_\mathcal{P}$. Thus, its projection onto DFA $\mathcal{A}_\varphi$ is a valid run of $\mathcal{A}_\varphi$. Thus a run that ends up in an accepting state in S_f corresponds to also an accepting run in $\mathcal{A}_\varphi$ and satisfies φ. Therefore, the problem reduces to finding a robot strategy $\sigma \in \Sigma^\mathcal{P}$ on $\mathcal{P}$ that guarantees that the robot can enforce a visit to the accepting set S_f under all possible human moves [3].

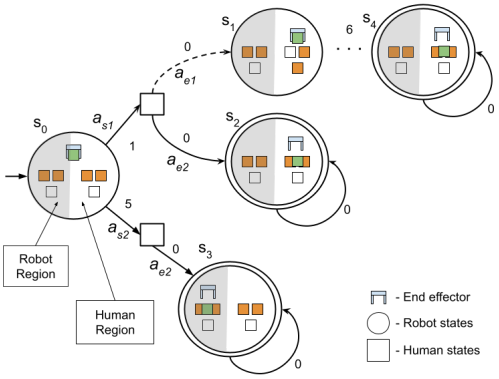


Fig. 3: DFA Game $\mathcal{P}$ for the abstraction $\mathcal{G}$ in Fig. 2 and the arch-building task. The double edged states denote accepting states. When operating close to the human, the robot spends either 1 (cooperative human) or 7 (adversarial human) units of total energy. Otherwise, it spends 5 units of total energy.

Example 4. *Fig. 3 illustrates a DFA Game for the abstraction $\mathcal{G}$ in Fig. 2 with the arch-building task. Say, the energy budget for the robot $\mathcal{B} = 7$. A strategy for the robot that guarantees task completion within the budget is to take action a_{s2} from s_0, which requires 5 units of energy, irrespective of the human's action. This strategy is obtained if human is assumed to be adversarial.*

Below, we show how to formulate $\mathcal{P}$ as a regret game to relax the adversarial assumption.

B. Reactive Synthesis with no Regret

An adversarial assumption is often a conservative abstraction of the reality because humans tend to be cooperative. Thus, we relax this assumption and also account for the fact that the human can have their own objective. Rather than picking an action which is optimal for all possible human actions, we pick an action that is "good enough" for all human actions. We propose regret to be a viable solution concept that quantifies how good an action is compared to the best possible action associated with the best outcome, i.e., the run with the least payoff in $\mathcal{P}$.

To build the intuition, consider the DFA Game in Fig. 3. The robot has two choices from the initial state s_0: to build the arch within human's reach (action a_{s_1}) or away from the human (action a_{s2}). The best action for the robot if the human decides to intervene is action a_{s2} and spend 5 units of total energy. If robot takes action a_{s1}, it spends 1 unit of energy for a cooperative human and 7 units of energy for an intervening human. Then, we say the "regret" associated with robot action a_{s1} is 2 if the human intervenes because the robot could have achieved the task with 5 units of energy with a_{s2}. If the human does not intervene, then regret of a_{s1} is 0. Similarly, if the robot decides to take action a_{s2} and the human decides to intervene, regret is 0, otherwise it is 4. Table I summarizes the regret values associated with all strategies. A regret-minimizing strategy for the robot is then to build the arch near the human (action a_{s1}) as it can guarantee a minimum regret of 2 under all human actions. Intuitively, a regret-minimizing strategy for this task is to

TABLE I: The regret associated with each robot and human action from (3) for DFA Game $\mathcal{P}$ in Fig. 3

Human (τ) \ Robot (σ)	near (a_{s1})	away (a_{s2})
Intervene (a_{e1})	7 - 5 = 2	5 - 5 = 0
No Intervene (a_{e2})	1 - 1 = 0	5 - 1 = 4

give the human a chance to be cooperative as it can still achieve the task with either actions while staying within the energy budget $\mathcal{B}$. We now formally define *regret*.

1) Regret: We define regret as the *difference* in the outcome (total payoff) associated with the current strategy and the best possible outcome, i.e., if the human is purely cooperative. We refer to the best possible outcome as *best-response*. Thus, we use best-responses as yardsticks to compare the quality of each robot strategy.

Definition 8 (Task-Aware Regret)**.** *Given strategies σ and τ on DFA Game $\mathcal{P}$ for the robot and human, respectively, task-aware regret at state s is defined as the difference in the payoff* $\mathrm{Val}^s(\sigma,\tau)$ *and the best possible payoff* $\min_{\sigma'} \mathrm{Val}^s(\sigma',\tau)$ *under the same human strategy τ, i.e.,*

$$\mathrm{reg}^s(\sigma,\tau) = \mathrm{Val}^s(\sigma,\tau) - \min_{\sigma'} \mathrm{Val}^s(\sigma',\tau). \tag{3}$$

Here σ' is any alternate strategy for the robot other than σ, i.e., $\sigma' \in \Sigma^{\mathcal{P}} \setminus \{\sigma\}$. We say that $\mathrm{reg}^s(\sigma) = +\infty$ if $\sigma' \in \emptyset$.

We use best-responses ($\min_{\sigma'} \mathrm{Val}^s(\sigma',\tau)$) to compare the quality of each robot action for a fixed τ of the human. As the regret behaviors change according to the task at hand, we call them task-aware regret. Hence, our goal is find a strategy σ^* that behaves "not far" from an optimal response to the strategy of the human τ when τ is fixed. Thus,

$$\sigma^* = \arg\min_{\sigma}(\max_{\tau} \mathrm{reg}^{\sigma,\tau}(s_0)). \tag{4}$$

In Table I, the combination of strategies (a_{s1}, a_{e2}) and (a_{s2}, a_{e1}) represent the extreme scenarios. A regret minimizing strategy σ^* is to pick a path with lower regret value - the worst-case payoff for current strategy σ is similar to the payoff of the best alternative σ'. Thus, we pick action a_{s1} as it has lower regret value. Note that the play induced by action a_{s1} also has a path, in which the human can be cooperative and help the robot spend less amount of energy to accomplish the task. Thus, the robot seeks for strategies that have similar best alternative payoffs from the current state. If the human is cooperative then the robot has more energy to spare and hence gives the human more opportunities to collaborate before finishing the task. The minimum budget $\mathcal{B}$ required to synthesize a regret-minimizing strategy is the total payoff associated with the winning strategy for the robot to complete the task assuming human to be purely adversarial.

We note that a key difference in our regret formulation from the ones used in the machine learning community is that we look at the set of all strategies, including the finite-memory ones for both the human and the robot. In the learning community, they use regret to compare the performance of their algorithms against a subset of strategies or a fixed adversary. For us, finite memory strategies are useful in keeping track of past human actions and thus allowing the robot to reason over the human's intention.

Algorithm 1: Compute regret-minimizing strategies

Input : DFA Game $\mathcal{P}$ and energy budget $\mathcal{B}$
Output: Regret-minimizing strategy σ^*
1 $\mathcal{G}^u \leftarrow$ `GraphOfUtility` ($\mathcal{P}$, $\mathcal{B}$)
2 $BA[e] \leftarrow \infty$ for all edges in $\mathcal{G}^u$
3 **for** *all* edges $\in \mathcal{G}^u$ **do**
4 $\quad BA[e] \leftarrow$ `ComputeBA` (edges)
5 **end**
6 $\mathcal{G}^{br} \leftarrow$ `GraphOfBestResponse` ($\mathcal{G}^u$, BA)
7 $\sigma^* \leftarrow$ `ValueIteration` ($\mathcal{G}^{br}$)
8 **return** σ^*

C. Computing Regret-Minimizing Strategies

To compute for σ^* in (4), we use the results in [13]. The method is summarized in Algorithm 1, which consists of two main steps. First, we compute all the plays induced by all robot strategies $\sigma \in \Sigma^{\mathcal{P}}$ for a fixed human strategy $\tau \in \mathrm{T}^{\mathcal{P}}$ by *unfolding* $\mathcal{P}$ until the payoff of $\mathcal{B}$ is reached. This process gives us a tree-like structure. We then augment each node with the value of the total energy spent by the robot (Val) to reach that node. We call this graph the *Graph of Utility* - $\mathcal{G}^u$ (Line 1). It captures all the possible plays for a fixed human strategy τ. Note that paths of $\mathcal{G}^u$ are realizations of finite memory strategies. Furthermore, if an accepting state in $\mathcal{P}$ is reached during unfolding, it is a leaf node in $\mathcal{G}^u$.

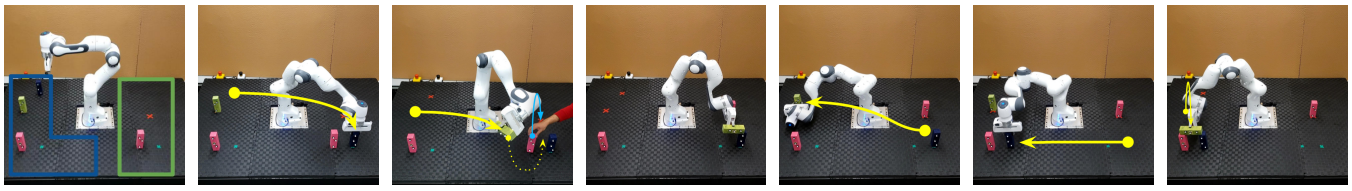
For each edge in every path in $\mathcal{G}^u$, we compute the best-alternate response (BA) by finding the lowest payoff that corresponds to $\min_{\tau} \min_{\sigma'} \mathrm{Val}^s(\sigma',\tau)$ (Line 4). We augment each node in $\mathcal{G}^u$ with the best-alternate response value and construct $\mathcal{G}^{br}$ (Line 6). For node $s \in \mathcal{G}^{br}$, the difference between its payoff and its best-alternate response is the regret value reg^s at s. We repeat this process for all nodes in $\mathcal{G}^{br}$. Then, we run a value iteration based method to back propagate the reg^s values from the leaf nodes in $\mathcal{G}^{br}$ until we reach a fixed point.

The number of states of $\mathcal{G}^{br}$ (denoted by $|\mathcal{G}^{br}|$) is polynomial in the size of $\mathcal{P}$. The algorithm is polynomial in the size of $\mathcal{P}$ times $\mathcal{B}$. The memory of strategies is directly related to the depth of the tree and the number of alternate edges along that path.

IV. Experiments

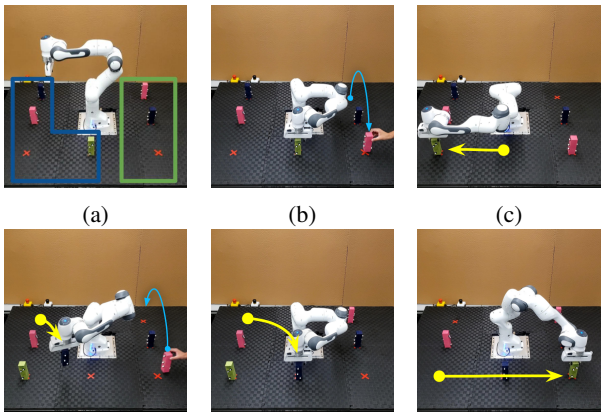
We illustrate the efficacy of our framework on two different case studies analogous to the ones in [2], [3]. In each scenario, the robot has two regions to complete the task. One is near the human, who may intervene or collaborate with the robot. The other region is far from the human, where they cannot reach. The robot spends 3 units of energy per action to operate away from human and 1 unit of energy near the human and was given a fixed budget $\mathcal{B}$ to complete the task.

We constructed an abstraction for each scenario as in [2], [3] (see [22] for details). Then, we used our regret-based



(a) (b) (c) (d) (e) (f) (g)

Fig. 4: Arch construction with one human intervention. Human and robot regions are indicated in green and blue respectively in (a). Yellow and blue arrow represent robot and human actions. Video: youtu.be/ABZb1g36Kv4



(a) (b) (c) (d) (e) (f)

Fig. 5: Straight line alignment with two human interventions. Robot (blue) and human (green) regions are shown in (a).

synthesis framework to generate a strategy and compared the emergent behaviors against the behaviors seen when using the framework in [2], which assumes the human is purely adversarial. Fig. 4 and 5 show emergent behaviors, and Table II outlines the computational costs. Videos of all the case-studies are available to view in [19].

The implementation of our framework is an end-to-end software tool that takes in the manipulation domain abstraction and task as a LTLf formula and generates a regret-minimizing strategy. The tool is in Python and is available on GitHub [20]. For the experiments, the strategy was implemented on the Franka Emika Panda robotic manipulator.

Arch Construction: In the first scenario, the task is to build an arch with the green box on top (LTLf formula shown in Example 3). The total budget was $\mathcal{B} = 10$, and the total energy needed to finish the task away from the human is 4 units. Fig. 5 shows an execution of this task with one human intervention. The robot initially starts to build the arch near the human as shown in Fig. 4a-4c. But, the human intervenes adversarially by moving the support (pink object) away. Then, the robot becomes conservative and builds the arch in the other region, spending 6 units of energy. A purely adversarial behavior for the robot is to build the arch away from the human, thus spending 4 units of total energy. Note that the robot ends up spending more energy under the regret-minimizing strategy, but it still stays within its energy budget while seeking collaboration. The general trend for such strategies is to be optimistic and seek collaboration with the human until the human disrupts the cooperation, motivating the robot to become pessimistic to ensure completing the task within the given energy budget.

Straight Line Alignment: The second task for the robot is to put three objects in a line such that pink block is in the top location, the blue block is in the middle, and the green block is at the bottom. Fig. 5 shows an execution of this task. Note that the robot needs to execute fewer actions to accomplish the task near the human. An adversarial strategy for the robot is to rearrange the blocks placed away from the human irrespective of the human's action. However, a regret minimizing strategy is to consider the possibility that the human could be cooperative. For this scenario $\mathcal{B}$ is 20 and the total energy to finish the task away from the human is 12, and 2 if the human is cooperative. Initially, the human intervenes, adversarially, thus the robot operates away from the human (Fig. 5a-5c). The human then intervenes again and opens up another opportunity for cooperation (Fig. 5d-5f). The robot takes this opportunity and completes the task.

Computational Cost: As the total energy required to accomplish the task without any human cooperation increases, we see that the robot provides more opportunities for the human to be cooperative as long as it can still guarantee to complete the task while staying below the energy budget $\mathcal{B}$. However, it comes at a cost. As shown in Table II, as $\mathcal{B}$ increases, more memory is required to generate the strategy. Therefore, it becomes computationally more expensive.

TABLE II: Total number of states in various abstractions, energy budget $\mathcal{B}$, and average runtime and memory usage.

Case study	$\lvert S_{\mathcal{P}} \rvert$	$\mathcal{B}$	$\lvert \mathcal{G}^{u} \rvert$	$\lvert \mathcal{G}^{br} \rvert$	Time (s)	Memory (GB)
Arch	48,843	10	537,274	2,152,851	405	6.12
		12	634,960	3,625,717	515	8.26
		14	732,721	5,477,123	622	12.10
		16	830,417	7,755,377	756	16.80
		18	928,113	10,469,395	898	21.88
Line	19,254	15	308,065	2,306,785	174	5.03
		17	346,573	3,263,221	221	7.01
		19	385,081	4,368,121	273	8.89
		21	423,589	5,621,485	328	11.72
		23	462,097	7,023,313	371	13.91

V. CONCLUSION

We presented a different formulation for synthesizing strategies for a robot operating in presence of a human. We use regret to relax the adversarial assumption on human and allow the robot to seek collaboration. We find the emergent behavior for the robot to be more intuitive and human-like. For future work, we plan to extend this framework to more complex scenarios with multiple agents as well as improving the computational cost of algorithm for efficient synthesis.

REFERENCES

[1] K. He, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "Towards manipulation planning with temporal logic specifications," in *IEEE international conference on robotics and automation*. IEEE, 2015, pp. 346–352.

[2] ——, "Reactive synthesis for finite tasks under resource constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2017, pp. 5326–5332.

[3] ——, "Automated abstraction of manipulation domains for cost-based reactive synthesis," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 285–292, 2019.

[4] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[5] C. I. Vasile and C. Belta, "Reactive sampling-based temporal logic path planning," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 4310–4315.

[6] E. M. Wolff, U. Topcu, and R. M. Murray, "Efficient reactive controller synthesis for a fragment of linear temporal logic," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 5033–5040.

[7] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, ser. IJCAI '13. AAAI Press, 2013, p. 854–860.

[8] A. M. Wells, Z. Kingston, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "Finite-horizon synthesis for probabilistic manipulation domains," in *IEEE International Conference on Robotics and Automation*. IEEE, 2021, (to appear).

[9] L. Chen, H. Luo, and C.-Y. Wei, "Minimax regret for stochastic shortest path with adversarial costs and known transition," in *Conference on Learning Theory*. PMLR, 2021, pp. 1180–1215.

[10] P. Jin, K. Keutzer, and S. Levine, "Regret minimization for partially observable deep reinforcement learning," in *International conference on machine learning*. PMLR, 2018, pp. 2342–2351.

[11] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret minimization in games with incomplete information," *Advances in neural information processing systems*, vol. 20, pp. 1729–1736, 2007.

[12] M. G. Azar, I. Osband, and R. Munos, "Minimax regret bounds for reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2017, pp. 263–272.

[13] E. Filiot, T. Le Gall, and J.-F. Raskin, "Iterated regret minimization in game graphs," in *International Symposium on Mathematical Foundations of Computer Science*. Springer, 2010, pp. 342–354.

[14] L. Chen and H. Luo, "Finding the stochastic shortest path with low regret: The adversarial cost and unknown transition case," *arXiv preprint arXiv:2102.05284*, 2021.

[15] J. Y. Halpern and R. Pass, "Iterated regret minimization: A new solution concept," *Games and Economic Behavior*, vol. 74, no. 1, pp. 184–207, 2012.

[16] P. Jin, K. Keutzer, and S. Levine, "Regret minimization for partially observable deep reinforcement learning," in *International conference on machine learning*. PMLR, 2018, pp. 2342–2351.

[17] P. Hunter, G. A. Pérez, and J.-F. Raskin, "Reactive synthesis without regret," *Acta informatica*, vol. 54, no. 1, pp. 3–39, 2017.

[18] ——, "Minimizing regret in discounted-sum games," *arXiv preprint arXiv:1511.00523*, 2015.

[19] K. Muvvala, 2021, online. [Online]. Available: https://youtu.be/ABZb1g36Kv4

[20] ——, "Regret based reactive synthesis," https://github.com/aria-systems-group/PDDLtoSim.

[21] P. Haslum, N. Lipovetzky, D. Magazzeni, and C. Muise, *An Introduction to the Planning Domain Definition Language*. Morgan & Claypool, 2019.

[22] K. Muvvala, "Human-aware strategy synthesis for robotic manipulators using regret games," M.S. thesis, University of Colorado at Boulder, 2021.