The Trembling-Hand Problem for LTL$_f$ Planning

Pian Yu[1], **Shufang Zhu**[1], **Giuseppe De Giacomo**[1], **Marta Kwiatkowska**[1] and **Moshe Vardi**[2]

[1]Department of Computer Science, University of Oxford, UK

[2] Department of Computer Science, Rice University, USA

{pian.yu, shufang.zhu, giuseppe.degiacomo, marta.kwiatkowska}@cs.ox.ac.uk
vardi@cs.rice.edu

Abstract

Consider an agent acting to achieve its temporal goal, but with a "trembling hand". In this case, the agent may mistakenly instruct, with a certain (typically small) probability, actions that are not intended due to faults or imprecision in its action selection mechanism, thereby leading to possible goal failure. We study the trembling-hand problem in the context of reasoning about actions and planning for temporally extended goals expressed in Linear Temporal Logic on finite traces (LTL$_f$), where we want to synthesize a strategy (aka plan) that maximizes the probability of satisfying the LTL$_f$ goal in spite of the trembling hand. We consider both deterministic and nondeterministic (adversarial) domains. We propose solution techniques for both cases by relying respectively on Markov Decision Processes and on Markov Decision Processes with Set-valued Transitions with LTL$_f$ objectives, where the set-valued probabilistic transitions capture both the nondeterminism from the environment and the possible action instruction errors from the agent. We formally show the correctness of our solution techniques and demonstrate their effectiveness experimentally through a proof-of-concept implementation.

1 Introduction

In this paper, we study the *trembling-hand* (TH) problem in the context of reasoning about actions and planning for temporally extended goals expressed in Linear Temporal Logic on finite traces LTL$_f$ [De Giacomo and Vardi, 2013].[1] In a chess game, a player may have a trembling hand due to, e.g., nervousness, anxiety, or stress, which results in mistaken moves that were not intended. Likewise, an agent acting in an environment could mistakenly instruct a different action, e.g., due to faults, leading to possible goal failure.

[1]All results presented here apply to other linear temporal logics on finite traces, such as LDL$_f$ [De Giacomo and Vardi, 2013] and Pure Past LTL [Bonassi *et al.*, 2023], as long as the set of traces that satisfy a formula can be characterized by a regular language, i.e., by a deterministic finite state automaton.

The TH problem originates from Game Theory in Economics, see e.g. [Marchesi and Gatti, 2021], referring to the situation where players erroneously select unintended moves with a small quantifiable probability. This problem highlights the importance of introducing some form of resilience to these errors [Vardi, 2020] in the player strategies and has given rise to the well-known notion of Trembling Hand Perfect Equilibrium in Economics [Bielefeld, 1988].

Here, we study this problem in the context of reasoning about actions [Reiter, 2001] and planning [Geffner and Bonet, 2013]. Specifically, we consider an agent acting in a domain. At each state of the domain, when the agent instructs an action, with a certain probability, it can mistakenly instruct a different action. Notice that this uncertainty is on the agent decision-making capabilities, not on how the environment executes the instructed actions. To stress this point, we consider two settings: deterministic domains, where the environment has no choices in responding to agent actions, as in classical planning [Geffner and Bonet, 2013]; and nondeterministic domains, where the environment can adversarially respond to agent actions, as in Fully Observable Nondeterministic Domains (FOND), when considering strong plans [Cimatti *et al.*, 2003; Ghallab *et al.*, 2004; Geffner and Bonet, 2013; De Giacomo and Rubin, 2018]. In both settings, we want to synthesize a strategy (aka plan) that guarantees to maximize the probability of fulfilling a temporally extended goal expressed in LTL$_f$, in spite of the adversarial response of the environment in the case of nondeterministic domains.

We devise solution techniques to solve the problem in the two settings by relying respectively on Markov Decision Processes (MDPs) [Puterman, 2014] in the case of deterministic domains and on Markov Decision Processes with Set-valued Transitions (MDPSTs) [Trevizan *et al.*, 2007; Trevizan *et al.*, 2008] in the case of nondeterministic domains. MDPs specify concrete probability values for each transition. MDPs with imprecise probabilities (MDPIPs) [White III and Eldeib, 1994; Satia and Lave Jr, 1973; Givan *et al.*, 2000] and Uncertain MDPs (UMDPs) [Nilim and El Ghaoui, 2004; Buffet *et al.*, 2005; Hahn *et al.*, 2019] have been proposed for scenarios where the probability values are uncertain. MDPSTs constitute a restricted subclass of MDPIPs, which combine probabilistically quantifiable uncertainty with unquantifiable uncertainty (nondeterminism) in a unified framework. They are attractive because they admit a simplified Bellman equation compared to MDPIPs, UMDPs [Trevizan *et al.*, 2007], and thus stochastic games.

In both settings, we consider LTL$_f$ objectives instead of standard reachability. Note that MDPs with LTL$_f$ objectives have been studied in [Brafman *et al.*, 2018; Wells *et al.*, 2021]. Instead, MDPSTs with LTL$_f$ objectives are studied for the first time in this paper. We lift the definition of satisfying LTL$_f$ objectives from MDPs to MDPSTs by defining the notion of robust strategy, and then an efficient value iteration algorithm is proposed for synthesizing an optimal robust strategy. We evaluate the effectiveness of the proposed solution techniques on a human-robot co-assembly problem, where the robot operates with a trembling hand, and demonstrate promising scalability.

1.1 Related Work

Interestingly, the trembling-hand problem has never been specifically studied in reasoning about actions and planning, though some related work exists. For example, the classical work on reasoning with noisy sensors and effectors in Situation Calculus [Bacchus *et al.*, 1999] is indeed related. There, sensors and effectors are considered to be noisy, so they introduce a stochastic element to be taken into account in reasoning. In particular, uncertainty on the effectors may be considered similarly to our trembling hand. There is, however, an important distinction between the two: in our case, the uncertainty is on which action is actually *instructed by the agent*; instead, in their case, the uncertainty is on how the action (perfectly instructed) is actually *executed by the environment*. In other words, in our case, the uncertainty is on the agent, while in theirs, the uncertainty is on the environment, which has been more extensively studied since it corresponds to uncertainty in modeling the reaction of the world to agent moves. In fact, there is a growing interest in forms of synthesis/planning that are resilient to errors in modeling the environment. For example, in [Zhu and De Giacomo, 2022], one could compute a maximally permissive strategy for the agent, allowing it to switch from one strategy to another while in execution, in case exceptional environment changes occur such that the predetermined strategy fails. In [Aminof *et al.*, 2021; Ciolek *et al.*, 2020], multiple models of the environment are considered to handle exceptions during planning, aiming to mitigate the intrinsic risk in a single environment model. In [Aineto *et al.*, 2023], a k-resilient strategy allows the agent to fail k times maximally at a repeatedly occurring state.

Nevertheless, in all these works, the focus is on errors/exceptions wrt expected environment behaviors. The trembling hand, on the other hand, is about errors in the agent behavior, and, as such, has not been much studied yet. Only two papers [Wells *et al.*, 2020a; Wells *et al.*, 2021] touched on this aspect, accounting for possible errors in robot decisions. Nevertheless, the environment considered is either deterministic or has probabilistic uncertainty. In contrast, our environment admits adversarial behaviors.

2 Preliminaries

We study the *trembling hand* (TH) problem in the context of planning for temporally extended goals expressed in LTL$_f$. We now briefly introduce the logic LTL$_f$, deterministic and nondeterministic planning domains, and the notion of strategy (aka plan) in a domain achieving an LTL$_f$ goal.

LTL$_f$. *Linear Temporal Logic on finite traces* (LTL$_f$) is a specification language expressing temporal properties on finite, nonempty traces. In particular, LTL$_f$ shares syntax with LTL, which is instead interpreted over infinite traces [Pnueli, 1977]. Given a set of atomic propositions $Prop$, LTL$_f$ formulas are generated as follows:
$$\varphi ::= a \mid \varphi \wedge \varphi \mid \neg\varphi \mid \circ\varphi \mid \varphi\, \mathcal{U}\, \varphi,$$
where $a \in Prop$ is an *atom*, $\circ$ (*Next*) and $\mathcal{U}$ (*Until*) are temporal operators. We make use of standard Boolean abbreviations, e.g., $\vee$ (or) and $\rightarrow$ (implies), $true$ and $false$. In addition, we define the following abbreviations: *Weak Next* $\bullet\varphi \equiv \neg\circ\neg\varphi$, *Eventually* $\Diamond\varphi \equiv true\, \mathcal{U}\, \varphi$ and *Always* $\Box\varphi \equiv false\, \mathcal{R}\, \varphi$, where $\mathcal{R}$ is for *Release*.

A *trace* $\pi = \pi_0\pi_1 \ldots$ is a sequence of propositional interpretations (sets), where for every $i \geq 0$, $\pi_i \in 2^{Prop}$ is the i-th interpretation of π. Intuitively, π_i is interpreted as the set of propositions that are $true$ at instant i. We denote the last instant (i.e., index) in a trace π by $\mathsf{lst}(\pi)$. By $\pi^k = \pi_0 \cdots \pi_k$ we denote the *prefix* of π up to the k-th iteration, and $\pi^k = \epsilon$ denotes an empty trace if $k < 0$. We denote π *satisfies* φ by $\pi \models \varphi$. The detailed semantics of LTL$_f$ can be found in [De Giacomo and Vardi, 2013]. It is also shown there that, for every LTL$_f$ formula φ, one can construct a Deterministic Finite Automaton (DFA) $Aut_\varphi = (2^{Prop}, \mathcal{Q}, q_0, \delta, acc)$, where 2^{Prop} is a finite alphabet, $\mathcal{Q}$ is a finite set of states, $q_0 \in \mathcal{Q}$ is the initial state, $\delta : \mathcal{Q} \times 2^{Prop} \rightarrow \mathcal{Q}$ is the transition function, and acc is the set of accepting states, such that for every trace π we have $\pi \models \varphi$ iff π is accepted by Aut_φ.

Deterministic Domain. A deterministic domain is a tuple $\mathcal{D} = (\mathcal{S}, s_0, A, F_d, \mathcal{L})$, where $\mathcal{S}$ is a finite set of states, $s_0 \in \mathcal{S}$ is an initial state, A is a finite set of *actions*, $F_d : \mathcal{S} \times A \mapsto \mathcal{S}$ is the deterministic transition function, where $s' = F_d(s, a)$ is the successor state after performing an *applicable* action a at s. We use $A(s) \subseteq A$ to denote the set of *applicable* actions at s. $\mathcal{L} : \mathcal{S} \rightarrow 2^{Prop}$ is the labeling function, where $Prop$ is a finite set of propositions. Note that, compared to typical formulations of domains in planning [Geffner and Bonet, 2013], we are assuming that more than one state can have the same evaluation of the propositions (fluents).

Nondeterministic Domain. A nondeterministic domain is a tuple $\mathcal{N} = (\mathcal{S}, s_0, A, F_n, \mathcal{L})$, where $\mathcal{S}, s_0, A$, and $\mathcal{L}$ are defined as in deterministic domains, and $F_n : \mathcal{S} \times A \mapsto 2^{\mathcal{S}}$ is now a nondeterministic transition function such that $F_n(s, a)$ denotes the non-empty set of possible successor states that follow an applicable action $a \in A(s)$ in s. It is worth noting that domains that are typically compactly represented, e.g., in Planning Domain Description Language (PDDL) [Haslum *et al.*, 2019], can be encoded using a number of bits that is logarithmic in the number of states.

LTL$_f$ Planning. A *path* of $\mathcal{D}$ (resp. $\mathcal{N}$) is a finite or infinite sequence of alternating states and actions $\rho = s_0a_0s_1a_1 \cdots$, ending with a state if finite, where s_0 is the initial state, and $F_d(s_i, a_i) = s_{i+1}$ (resp. $s_{i+1} \in F_n(s_i, a_i)$) for all i with $0 \leq i < |\rho|$, and $|\rho| \in \mathbb{N} \cup \{\infty\}$. We denote by $\rho^k =$

$s_0a_0s_1a_1\cdots s_k$ the finite prefix of ρ up to the k-th alternation. The sequence $\pi(\rho,\mathcal{D}) = \mathcal{L}(s_0)\mathcal{L}(s_1)\cdots$ (resp. $\pi(\rho,\mathcal{N}) = \mathcal{L}(s_0)\mathcal{L}(s_1)\cdots$) over $Prop$ is called the *trace* induced by ρ over $\mathcal{D}$ (resp. $\mathcal{N}$). FPaths denotes the set of all finite paths. An *agent strategy* (or plan) is a function $\sigma_p : \mathrm{FPaths} \to A$ mapping a finite path on $\mathcal{D}$ (resp. $\mathcal{N}$) to agent actions that are applicable at the last state of the finite path. For nondeterministic domains, we assume that the nondeterminism is resolved by the environment, which acts according to an (unknown) strategy as an (adversarial) antagonist of the agent. Environment strategies are functions of the form $\gamma_p : \mathrm{FPaths} \times A \to \mathcal{S}$, which need to comply with the domain, in the sense that given a finite path $\rho \in \mathrm{FPaths}$ and an action $a \in A(\mathsf{lst}(\rho))$ it must be the case that $\gamma_p(\rho) \in F_n(\mathsf{lst}(\rho), a)$. Note that, in the case of deterministic domains, this constraint forces the environment to have only one strategy. Given an agent strategy σ_p and an environment strategy γ_p, there is a unique path $\rho(\sigma_p,\gamma_p) = s_0a_0s_1a_1\cdots$ generated by σ_p and γ_p, where s_0 is the initial state and for every $i \geq 0$ it holds that $a_i = \sigma_p(\rho^i)$ and $s_{i+1} = \gamma_p(\rho^i, a_i)$. Sometimes, for simplicity, we write ρ instead of $\rho(\sigma_p,\gamma_p)$, when it is clear in the context.

An agent strategy σ_p enforces an LTL$_f$ φ in a domain $\mathcal{D}$ (resp. $\mathcal{N}$) if for every environment strategy γ_p, the infinite path $\rho(\sigma_p,\gamma_p)$ contains a finite prefix ρ^i such that the finite trace $\pi(\rho^i,\mathcal{D}) \models \varphi$ (resp. $\pi(\rho^i,\mathcal{N}) \models \varphi$). LTL$_f$ planning concerns computing such an agent strategy σ_p, if one exists.

3 TH in Deterministic Domains

We begin investigating the *trembling-hand* problem for LTL$_f$ planning by focusing on the case of deterministic domains, where the environment has only one strategy, i.e., following the transitions of the domain. Hence, the only uncertainty is the stochastic one, arising from the "trembling hand". In the next section, we consider the case of nondeterministic domains, where the environment employs adversarial strategies.

3.1 Problem Formulation

The "trembling hand" refers to the agent intending to instruct a certain action but, by mistake, instructing a different action with a (small) quantified probability. This probability only depends on the domain state where the action is instructed (and then performed). Notice that, here, the environment is fully deterministic, and hence, once the action is instructed, it will be executed (without any error) in a deterministic way. The problem that we want to address is to maximize the probability of achieving a given LTL$_f$ objective in a deterministic domain in spite of the (state dependent) action-instruction errors due to the "trembling hand".

We formalize the action-instruction errors as follows. Let $\mathcal{D} = (\mathcal{S}, s_0, A, F_d, \mathcal{L})$ be a deterministic domain, $s \in \mathcal{S}$ a domain state, and $a \in A(s)$ an applicable action at s. We denote by $err(s,a) \in \mathrm{Dist}(A(s))$ the probability distribution of instructing an action a' instead of a in state s. For instance, suppose the set of applicable actions at s is such that $A(s) = \{a, a', a''\}$ and $err(s,a) = [0.9, 0.04, 0.06]$. This means that, when the agent intends to instruct action a at state s, then with probability 0.04 and 0.06, it may instruct actions a' and a'', respectively. Let $\mathcal{E} = \{err(s,a) : (s,a) \in \mathcal{S} \times A\}$ be the set of the (state-dependent) action-instruction errors caused by the "trembling hand". The set of actions that could be instructed when the agent intends to instruct action a at state s is the *support set* of $err(s,a)$, denoted by $supp(s,a) \subseteq A(s)$. In this example, $supp(s,a) = \{a, a', a''\}$.

Recall that, without a trembling hand, executing an agent strategy σ_p in a deterministic domain $\mathcal{D}$ results in a unique path. Yet, in the presence of a "trembling hand", i.e., action-instruction errors $\mathcal{E}$, we get perturbed paths, where the actually instructed action may differ from the intended one with a probability following $\mathcal{E}$.

Definition 1 (Perturbed path in $\mathcal{D}$)**.** *Let $\mathcal{D}$ be a deterministic domain, σ_p an agent strategy, and $\mathcal{E}$ a set of action-instruction errors. A* perturbed path *in $\mathcal{D}$ wrt $\mathcal{E}$ is a sequence of triples $\rho(\sigma_p,\mathcal{E}) = (s_0, a_0, a'_0)(s_1, a_1, a'_1)\cdots$, where for every $i \geq 0$, $a_i = \sigma_p(\rho^i)$ (the intended action), $a'_i \in supp(s_i, a_i)$ (the actually instructed action), and $s_{i+1} = F_d(s_i, a'_i)$. The set of all perturbed paths in $\mathcal{D}$ wrt σ_p and $\mathcal{E}$ is denoted by $\Phi^{\sigma_p,\mathcal{E}}$.*

Given a deterministic domain $\mathcal{D}$, an LTL$_f$ formula φ, and a set of action-instruction errors $\mathcal{E}$, the probability of σ_p enforcing φ in $\mathcal{D}$ wrt $\mathcal{E}$ is defined as $\mathrm{Pr}^{\sigma_p,\mathcal{E}}_{\mathcal{D}}(\varphi) := \mathrm{Pr}_{\mathcal{D}}(\{\rho \in \Phi^{\sigma_p,\mathcal{E}} \mid \pi(\rho^k,\mathcal{D}) \models \varphi \text{ for some } k \geq 0\})$.

Definition 2 (TH problem for LTL$_f$ planning in $\mathcal{D}$)**.** *The problem is a tuple $\mathcal{P}^d = (\mathcal{D}, \varphi, \mathcal{E})$, where $\mathcal{D}$ is a deterministic domain, φ is an LTL$_f$ formula, and $\mathcal{E}$ is a set of action-instruction errors. Solving $\mathcal{P}^d$ consists in synthesizing an agent strategy σ^*_p that maximizes the probability of enforcing φ in $\mathcal{D}$ wrt $\mathcal{E}$, i.e., an* optimal strategy *for $\mathcal{P}^d$, that is: $\sigma^*_p = \arg\max_{\sigma_p} \mathrm{Pr}^{\sigma_p,\mathcal{E}}_{\mathcal{D}}(\varphi)$.*

In the following, an example is given to demonstrate a perturbed transition in a deterministic domain $\mathcal{D}$.

Example 1. *Consider a robot assembly problem, where the robot aims to assemble an arch using N blocks. In Figure 1, the goal configurations for N $(2 \leq N \leq 6)$ blocks are depicted (which are used later in Section 5). Let $OBJ = \{Obj_i \mid i \in \{1,\cdots,N\}\}$ be the set of blocks and $LOC = \{L_j \mid j \in \{0,\cdots,M\}\}$ be the set of locations, where L_0 represents the storage. Initially, all the blocks are stored in the storage.*

During assembly, the robot can perform move *actions to relocate blocks. The set of robot actions is $A = \{(Obj_i, L_j) : Obj_i \in OBJ, L_j \in LOC\} \cup \{do\text{-}nothing\}$, where (Obj_i, L_j) means move block i to location j. Due to the "trembling hand" (which may be caused by drifting), the robot's action is subject to uncertainty. For instance, if the robot intends to move block i to location j, there exists a probability that it may mistakenly move a different block $i' \neq i$ or inaccurately place it in location $j' \neq j$. In addition, the probability of errors varies for different actions. For instance, if the robot chooses* DO-NOTHING*, one can safely assume that the probability of error is 0. However, if the robot chooses to* move*, we assume a positive probability of error, leading to perturbed transitions resulting in perturbed paths.*

Figure 2 shows a perturbed transition example in this case. The dashed arrow shows the intended action and the solid *arrows represent the set of actions that may be actually instructed with their respective probabilities.*

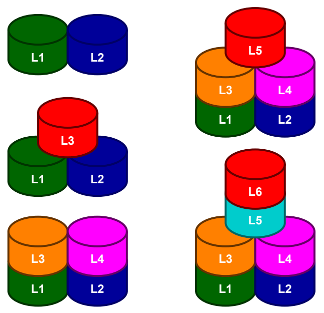

Figure 1: An arch. Left: 2, 3, and 4 blocks. Right: 5 and 6 blocks.

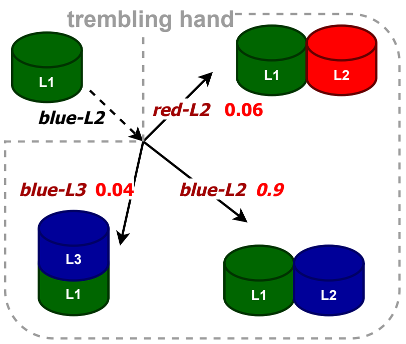

Figure 2: A perturbed transition in a det. domain.

3.2 MDPs With LTL$_f$ Objectives

Solving $\mathcal{P}^d = (\mathcal{D}, \varphi, \mathcal{E})$ requires a modeling technique that is able to capture the action-instruction errors $\mathcal{E}$ in a deterministic domain. To this end, we first review *Markov Decision Processes (MDPs)*, which are used in our solution technique.

MDPs allow action choice in each state, and each state action transition is a probability distribution on successor states, which provides a natural way of capturing the probability of the agent mistakenly taking a different action in a deterministic domain. Following [Puterman, 2014], an MDP is a tuple $\mathcal{M} = (\mathcal{S}, s_0, A, \mathcal{T}, \mathcal{L})$, where $\mathcal{S}$ is a finite set of states, $s_0 \in \mathcal{S}$ is the initial state, A is a finite set of actions, $\mathcal{T} : \mathcal{S} \times A \times \mathcal{S} \to [0, 1]$ is the probabilistic transition function, and $\mathcal{L} : \mathcal{S} \to 2^{Prop}$ is the proposition labelling function. Analogously to planning domains, for each $s \in \mathcal{S}$, the set of actions *applicable* at state s is denoted by $A(s)$.

Denote by $\xi = s_0 a_0 s_1 a_1 \cdots$ a path and by FPaths the set of all finite paths of $\mathcal{M}$. In this work, we focus on *deterministic* agent strategies for MDPs, instead of *randomized*. A strategy σ_m of $\mathcal{M}$ is a function σ_m : FPaths $\to A$ such that, for each $\xi \in$ FPaths, $\sigma_m(\xi) \in A(\mathsf{lst}(\xi))$, where $\mathsf{lst}(\xi)$ is the last state of ξ. We denote by Ξ^{σ_m} the set of all probably infinite paths of $\mathcal{M}$ generated by σ_m.

Given an MDP $\mathcal{M} = (\mathcal{S}, s_0, A, \mathcal{T}, \mathcal{L})$ and a set of goal states $G \subseteq \mathcal{S}$, the probability of an agent strategy σ_m enforcing G in $\mathcal{M}$ is defined as $\Pr^{\sigma_m}_{\mathcal{M}}(G) := \Pr_{\mathcal{M}}(\{\xi \in \Xi^{\sigma_m} \mid \mathsf{lst}(\xi^k) \in G \text{ for some } k \geq 0\})$. Computing an optimal strategy σ^*_m that maximizes the probability of enforcing G is the *reachability problem* over $\mathcal{M}$. Analogously, an agent strategy can also enforce a temporal objective on an MDP. Given an MDP $\mathcal{M}$ and an LTL$_f$ formula φ, the probability of an agent strategy σ_m enforcing φ in $\mathcal{M}$ is defined as $\Pr^{\sigma_m}_{\mathcal{M}}(\varphi) := \Pr_{\mathcal{M}}(\{\xi \in \Xi^{\sigma_m} \mid \pi(\xi^k, \mathcal{M}) \models \varphi \text{ for some } k \geq 0\})$. The problem of MDP with LTL$_f$ objective is to compute an optimal strategy σ^*_m, which maximizes the probability of enforcing φ in $\mathcal{M}$ [Baier and Katoen, 2008].

3.3 Solution Technique

We now present our solution technique to synthesize an optimal agent strategy σ^*_p for $\mathcal{P}^d = (\mathcal{D}, \varphi, \mathcal{E})$, which aims to maximize the probability of enforcing φ in $\mathcal{D}$ in spite of $\mathcal{E}$. The key idea is to reduce $\mathcal{P}^d$ to an MDP with an LTL$_f$ objective. Intuitively, the MDP provides a probabilistic abstraction of instructing mistaken actions in the domain due to the trembling hand. Hence, the MDP has the same states as the domain, retains its original action choices, and incorporates the probability of action-instruction errors in its transitions.

Probabilistic abstraction. We define an MDP $\mathcal{M} = (\mathcal{S}, s_0, A, \mathcal{T}, \mathcal{L})$ from $\mathcal{P}^d = (\mathcal{D}, \varphi, \mathcal{E})$ as follows. $\mathcal{S}, s_0, A$, and $\mathcal{L}$ are the same as in $\mathcal{D}$. In order to construct the probabilistic transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, note that $\mathcal{T}(s, a, s')$ gives the probability of transitioning from state s to s' on action a. Assuming no "trembling hand" errors, we have that $\mathcal{T}(s, a, s') = 1$ if $s' = F_d(s, a)$ and $\mathcal{T}(s, a, s') = 0$ otherwise. Due to "trembling-hand" errors, however, unintended actions (e.g., a') may be instructed with some probability (given by $err(s, a)(a')$), thus resulting in a different successor state $s'' = F_d(s, a')$. Following this observation, we construct $\mathcal{T}$ as follows:

$$\mathcal{T}(s, a, s') = \begin{cases} err(s, a)(a'), & \text{if } a' \in A(s), s' = F_d(s, a'), \\ 0, & \text{otherwise.} \end{cases}$$

The TH problem for LTL$_f$ planning in deterministic domains is now reduced to an MDP with an LTL$_f$ objective.

Theorem 1. *Let $\mathcal{P}^d = (\mathcal{D}, \varphi, \mathcal{E})$ be a TH problem defined in Def. 2, and $\mathcal{M}$ the constructed MDP described above. An optimal strategy for $\mathcal{M}$ with φ is an optimal strategy for $\mathcal{P}^d$ and vice versa, that is:*

$$\sigma^*_p = \arg\max_{\sigma_m} \{\Pr^{\sigma_m}_{\mathcal{M}}(\varphi)\},$$

where σ^*_p (optimal strategy for $\mathcal{P}^d$) is given in Def. 2.

Proof. First, we observe that $\sigma_p = \sigma_m$, by construction. Second, one can derive, according to Def. 1 that $\Phi^{\sigma_p, \mathcal{E}} = \Xi^{\sigma_m}$ if $\sigma_p = \sigma_m$. Finally, given a state s and an intended action $a = \sigma_p(s)$ in $\mathcal{P}^d$, one has that, with probability $err(s, \sigma_p(s))(a')$, the agent may actually instruct a different action a' and then transit to $s' = F_d(s, a')$. Denote by $\Pr_{\mathcal{P}^d}(s'|s, \sigma_p(s))$ the probability of transiting to state s' from the state s with action $\sigma_p(s)$. We obtain that if $\sigma_p(s) = \sigma_m(s)$, then $\Pr_{\mathcal{P}^d}(s'|s, \sigma_p(s)) = err(s, \sigma_p(s))(a') = \mathcal{T}(s, \sigma_m(s), s')$. Hence, the conclusion follows. $\square$

Thm. 1 allows us to utilize existing algorithms for MDPs with LTL$_f$ objectives to solve the TH problem for LTL$_f$ planning in deterministic domains. The common approach is by reduction to the reachability problem of an MDP [Baier and Katoen, 2008; Wells *et al.*, 2021]. More specifically, given an MDP $\mathcal{M}$ with LTL$_f$ objective φ, we first construct the corresponding DFA Aut_φ of the LTL$_f$ formula φ, then construct the product MDP $\mathcal{M}^\times$ of $\mathcal{M}$ and Aut_φ. In this case, computing an optimal strategy for $\mathcal{M}$ with φ reduces to the reachability problem over $\mathcal{M}^\times$, where the goal states G are those in $\mathcal{M}^\times$ that consist of the accepting states of Aut_φ. The reachability problem of $\mathcal{M}^\times$ wrt G can be solved via value iteration, strategy iteration, or linear programming [Altman, 1998; Wells *et al.*, 2021]. Due to the cross product, every finite path $\rho \in \mathrm{FPaths}_\mathcal{D}$ such that $\rho = s_0a_0s_1a_1\cdots s_{k+1}$, where $\mathrm{FPaths}_\mathcal{D}$ denotes the finite paths on $\mathcal{D}$, corresponds to a finite path $\rho^\times$ on $\mathcal{M}^\times$, where $\rho^\times = (s_0,q_0)a_0(s_1,q_1)a_1\cdots(s_k,q_k)a_k(s_{k+1},q_{k+1})$. Therefore, every strategy σ_m^* for $\mathcal{M}^\times$ wrt G induces a strategy for $\mathcal{P}^d = (\mathcal{D},\varphi,\mathcal{E})$ as follows: $\sigma_p^*(\rho) = \sigma_m^*(\rho^\times)$ for $\rho \in \mathrm{FPaths}_\mathcal{D}$. Together with Thm. 1, the following theorem is an immediate result.

Theorem 2. *Let $\mathcal{P}^d = (\mathcal{D},\varphi,\mathcal{E})$ be a TH problem defined in Def. 2, $\mathcal{M}^\times = \mathcal{M} \times Aut_\varphi$ the constructed product MDP, and G the set of goal states. We have that the computed optimal strategy σ_m^* of $\mathcal{M}^\times$ with reachability goal G induces an optimal strategy σ_p^* for $\mathcal{P}^d$.*

4 TH in Nondeterministic Domains

We now turn to the case in which the domain is nondeterministic, i.e., the environment can choose its strategy adversarially. Hence, in planning, we have to overcome two forms of uncertainty: the stochastic uncertainty from the trembling hand and the adversarial uncertainty from the environment.

4.1 Problem Formulation

Consider the case where the agent acts in a nondeterministic (adversarial) domain. We assume the nondeterminism is unquantifiable, so it is adversarial without a probabilistic behavior. In this setting, we want to synthesize an agent strategy that maximizes the probability of achieving its goal in spite of the adversarial behavior of the environment and the (small) probability of instructing wrong actions at every step. In other words, the agent seeks a maxi-min strategy, that is, a strategy that maximizes the minimal probability across all possible environment strategies. We find this case to be particularly interesting, since it combines the probabilistic aspects of the previous case's action-instruction errors and the environment's adversarial nondeterminism.

We first define perturbed paths of an agent acting in a nondeterministic domain $\mathcal{N}$ with action-instruction errors $\mathcal{E}$, i.e., the trembling hand, and adversarial environment behaviors.

Definition 3 (Perturbed paths in $\mathcal{N}$)**.** *Let $\mathcal{N}$ be a nondeterministic domain, σ_p an agent strategy, γ_p an environment strategy, and $\mathcal{E}$ a set of action-instruction errors. A* perturbed path *in $\mathcal{N}$ wrt $\mathcal{E}$ is a sequence of triples $\rho(\sigma_p,\gamma_p,\mathcal{E}) = (s_0,a_0,a_0')(s_1,a_1,a_1')\cdots$. We denote by ρ' the projection of ρ by considering only the states s_i and actually instructed actions a_i'. It holds that, for every $i \geq 0$, $a_i \in \sigma_p(\rho'^i)$ (the intended action), $a_i' \in supp(s_i,a_i)$ (the actually instructed action), and $s_{i+1} = \gamma_p(\rho'^i, a_i')$. The set of all perturbed paths in $\mathcal{N}$ wrt $\sigma_p,\gamma_p,\mathcal{E}$ is denoted by $\Phi^{\sigma_p,\gamma_p,\mathcal{E}}$.*

The probability of an agent strategy σ_p enforcing φ considering environment strategy γ_p and the action-instruction error $\mathcal{E}$ is denoted by $\Pr_\mathcal{N}^{\sigma_p,\gamma_p,\mathcal{E}}(\varphi)$.

Definition 4 (TH problem for LTL$_f$ planning in $\mathcal{N}$)**.** *The problem is a tuple $\mathcal{P}^n = (\mathcal{N},\varphi,\mathcal{E})$, where $\mathcal{N}$ is a nondeterministic domain, φ is an LTL$_f$ formula, and $\mathcal{E}$ is a set of action-instruction errors. Solving $\mathcal{P}^n$ consists in synthesizing an agent strategy σ_p^* that maximizes the probability of enforcing φ in $\mathcal{N}$ with $\mathcal{E}$ in spite of adversarial strategies of the environment, i.e., an* optimal strategy *for $\mathcal{P}^n$, that is:*

$$\sigma_p^* = \arg\max_{\sigma_p}\min_{\gamma_p} \Pr_\mathcal{N}^{\sigma_p,\gamma_p,\mathcal{E}}(\varphi).$$

Example 2. *Consider the human-robot co-assembly problem adapted from [He* et al.*, 2019], where the robot aims to assemble an arch along with a human in a shared workspace. During assembly, the robot can perform actions to* relocate *blocks, and the human may perform moves to intervene. The robot has a "trembling hand" as described in Example 1. Compounding this issue, the robot must ensure that the arch is successfully built, considering the human involvement.*

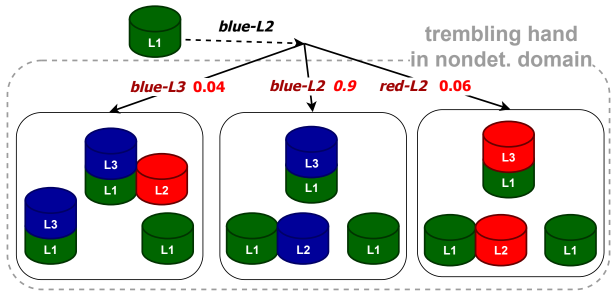

Figure 3: A perturbed transition in a nondet. domain.

At each step, the robot instructs an action to relocate a block, though with a trembling hand, and then the human can react by moving blocks among locations. Human movements are not controllable, thus introducing nondeterminism into the consequences of instructed robot actions. Figure 3 shows a perturbed transition example of a robot working in such a nondeterministic domain with a trembling hand. Note that each intended action (dashed arrow) corresponds to a set of actions (solid arrow) that may be actually instructed, together with their respective probabilities. Furthermore, each actually instructed action leads to a set of possible changes due to uncontrollable human intervention.

4.2 MDPSTs With LTL$_f$ Objectives

We now introduce *Markov Decision Processes with Set-valued Transitions (MDPSTs)*, which combine probabilistically quantifiable uncertainty with unquantifiable uncertainty (nondeterminism), hence providing a natural probabilistic abstraction of an agent acting with a trembling hand in a nondeterministic domain. Note that MDPSTs are favorable due to their simplified Bellman equation compared to MD-

PIPs, UMDPs, and thus stochastic games. This distinction will be further clarified in Section 4.3.

An *MDPST* is a tuple $\mathcal{M}_N = (\mathcal{S}, s_0, A, \mathcal{F}, \mathcal{T}_N, \mathcal{L})$, where $\mathcal{S}, s_0, A$, and $\mathcal{L}$ are defined as for MDPs, and

- $\mathcal{F} : \mathcal{S} \times A \Rightarrow 2^{\mathcal{S}}$ is the set-valued nondeterministic transition function;
- $\mathcal{T}_N : \mathcal{S} \times A \times 2^{\mathcal{S}} \to [0,1]$ is the transition probability (or mass assignment) function, i.e., given a set $\Theta \in \mathcal{F}(s,a)$, where $\Theta \subseteq \mathcal{S}$, $\mathcal{T}_N(s,a,\Theta) = \Pr(\Theta|s,a)$.

Note that, in MDPSTs, the transition function $\mathcal{F}(s,a)$ returns a set of state sets, i.e., $\mathcal{F}(s,a) \subseteq 2^{\mathcal{S}}$, and the transition probability function $\mathcal{T}$ expresses the probability of transitioning to such sets via a given action.

It is worth noting that the problem of MDPSTs with LTL$_f$ objectives is studied for the first time in this paper. To define the problem, one needs to lift the definition of enforcing LTL$_f$ objectives from MDPs to MDPSTs by defining the notion of robustness for strategies to incorporate the unquantifiable uncertainty. In an MDPST, due to the unquantifiable uncertainty, the distribution on the set of reachable successor states of a state-action pair (s,a), where $a \in A(s)$, is not uniquely determined by (s,a), in contrast to MDPs. We denote by $\mathrm{Post}(s,a)$ the set of reachable states of (s,a), despite both quantifiable and unquantifiable uncertainties. Formally, $\mathrm{Post}(s,a) = \{s' \mid \exists \Theta \in \mathcal{F}(s,a) \text{ s.t. } \mathcal{T}_N(s,a,\Theta) > 0$ and $s' \in \Theta\}$. A *feasible distribution* of an MDPST guarantees that, given a state action pair (s,a), where $a \in A(s)$, *(i)* only one state is chosen within Θ for each $\Theta \in \mathcal{F}(s,a)$; *(ii)* the sum of probabilities of selecting a state from $\mathrm{Post}(s,a)$ equals 1; and *(iii)* the probability of selecting a state out of $\mathrm{Post}(s,a)$ is 0. In the following definition, $\iota_{s'}^{\Theta}$ indicates whether s' is *in* Θ. Hence $\iota_{s'}^{\Theta} = 1$ if $s' \in \Theta$ and $\iota_{s'}^{\Theta} = 0$ otherwise. Furthermore, $\alpha_{s'}^{\Theta}$ indicates whether s' is *selected* from Θ. Hence $\alpha_{s'}^{\Theta} = 1$ if s' is selected from Θ, and $\alpha_{s'}^{\Theta} = 0$ otherwise.

Definition 5 (Feasible distribution in MDPSTs)**.** *Let $\mathcal{M}_N = (\mathcal{S}, s_0, A, \mathcal{F}, \mathcal{T}_N, \mathcal{L})$ be an MDPST, (s,a) a state-action pair, where $a \in A(s)$. $\mathfrak{h}_s^a \in \mathrm{Dist}(\mathcal{S})$ is a* feasible distribution *of (s,a), denoted by $s \xrightarrow{a} \mathfrak{h}_s^a$, if*

(i) $\sum_{s' \in \Theta} \alpha_{s'}^{\Theta} = 1$, *for* $\Theta \in \mathcal{F}(s,a)$*;*

(ii) $\mathfrak{h}_x^a(s') = \sum_{\Theta \in \mathcal{F}(s,a)} \iota_{s'}^{\Theta} \alpha_{s'}^{\Theta} \mathcal{T}_N(s,a,\Theta)$, *for* $s' \in \mathrm{Post}(s,a)$*;*

(iii) $\mathfrak{h}_s^a(s') = 0$, *for* $s' \in \mathcal{S} \setminus \mathrm{Post}(s,a)$.

Following Def. 5, it is evident that, in MDPSTs, a feasible distribution is not uniquely determined for a given state-action pair, in contrast to MDPs. It highly depends on $\alpha_{s'}^{\Theta}$ (see item (ii)), the unquantifiable uncertainty of whether s' is selected from Θ. We now introduce *nature* for MDPSTs to characterize this unquantifiable uncertainty, motivated by the definition of *nature* in robust MDPs [Nilim and El Ghaoui, 2004]. One can intuitively perceive nature as the environment in the context of nondeterministic domains, playing a role in resolving nondeterminism. We denote by $\mathcal{H}_s^a$ the set of feasible distributions of state action pair (s,a). Analogously to MDPs, FPaths denotes the set of finite paths of an MDPST.

Definition 6 (Nature for MDPSTs)**.** *A* nature *of an MDPST is a function $\gamma_m : \mathrm{FPaths} \times A \to \mathrm{Dist}(\mathcal{S})$ such that, $\gamma(\xi, a) \in \mathcal{H}_s^a$ for $\xi \in \mathrm{FPaths}$ and $a \in A(\mathsf{lst}(\xi))$.*

Suppose we fix a nature γ_m. The probability of an agent strategy σ_m enforcing φ is denoted by $\Pr_{\mathcal{M}_N}^{\sigma_m,\gamma_m}(\varphi) := \Pr_{\mathcal{M}_N}(\{\xi \in \Xi^{\sigma_m,\gamma_m} \mid \pi(\xi^k, \mathcal{M}_N) \models \varphi \text{ for some } k \geq 0\})$, where Ξ^{σ_m,γ_m} is the set of all probable paths generated by the agent strategy σ_m and nature γ_m. We now define (optimal) robust strategies for MDPSTs, which quantify all natures.

Definition 7 (Robust strategy)**.** *Let $\mathcal{M}_N$ be an MDPST, φ an LTL$_f$ formula, and $\beta \in [0,1]$ a threshold. An agent strategy σ_m* robustly enforces *φ in $\mathcal{M}_N$ wrt β, if for every nature γ_m, the probability of the probably generated paths satisfying φ is no less than β, that is, $P_{\mathcal{M}_N}^{\sigma_m}(\varphi) \geq \beta$, where $P_{\mathcal{M}_N}^{\sigma_m}(\varphi) := \min_{\gamma_m}\{\Pr_{\mathcal{M}_N}^{\sigma_m,\gamma_m}(\varphi)\}$. Such σ_m is referred to as a robust strategy for $\mathcal{M}_N$ (wrt β).*

Definition 8 (Optimal robust strategy)**.** *An optimal strategy σ_m^* robustly enforces an LTL$_f$ formula φ in an MDPST $\mathcal{M}_N$ is $\sigma_m^* = \arg\max_{\sigma_m}\{\Pr_{\mathcal{M}_N}^{\sigma_m}(\varphi)\}$. In this case, σ_m^* is referred to as an* optimal robust strategy *for $\mathcal{M}_N$.*

The problem of MDPSTs with LTL$_f$ objectives is to compute an optimal robust strategy σ_m^*, which maximizes the probability of robustly enforcing φ in $\mathcal{M}_N$, i.e., achieving the maximal value of β. Analogously, we can also define an MDPST with simple reachability, i.e., reaching a set of goal states. To avoid repetition, it has been omitted.

4.3 Solution Technique

The key idea to solve the TH problem for LTL$_f$ planning in nondeterministic domains is to combine the quantifiable action-instruction errors and the unquantifiable adversarial nondeterminism of the domain into an MDPST. This MDPST has the same states as the domain $\mathcal{N}$, and incorporates the action-instruction errors and the domain's adversarial nondeterminism into its transitions. In this case, we reduce $\mathcal{P}^n$ to an MDPST with an LTL$_f$ objective.

Probabilistic abstraction. We build an MDPST $\mathcal{M}_N = (\mathcal{S}, s_0, A, \mathcal{F}, \mathcal{T}_N, \mathcal{L})$ from $\mathcal{P}^n = (\mathcal{N}, \varphi, \mathcal{E})$ as follows. S, s_0, A, and $\mathcal{L}$ are the same as in $\mathcal{N}$. To construct the set-valued nondeterministic transition function $\mathcal{F}$ and the mass assignment function $\mathcal{T}_N$, we incorporate the action-instruction errors $\mathcal{E}$ into the probabilistic transitions, much like in deterministic domains. In nondeterministic domains, however, the successor for each state-action pair is not singular due to the adversarial environment behaviour. Consequently, the successor of each probabilistic transition is a set, with environment deciding which element of the set to transit to.

Based on these observations, we can construct the set-valued nondeterministic transition function $\mathcal{F}$ as $\mathcal{F}(s,a) = \cup_{a' \in A(s)}\{F_n(s,a')\}$, such that $\mathcal{F}(s,a)$ is a set of subsets in $\mathcal{S}$, for $s \in \mathcal{S}$ and $a \in A(s)$. The corresponding mass assignment function $\mathcal{T}_N$ is such that

$$\mathcal{T}_N(s,a,\Theta) = \begin{cases} err(s,a)(a'), & \text{if } a' \in A(s), \Theta = F_n(s,a'), \\ 0, & \text{otherwise.} \end{cases}$$

We now reduce $\mathcal{P}^n$ to an MDPST with an LTL$_f$ objective.

Theorem 3. *Let $\mathcal{P}^n = (\mathcal{N}, \varphi, \mathcal{E})$ be a problem defined in Def. 4, and $\mathcal{M}_N$ the constructed MDPST described above. An optimal robust strategy for $\mathcal{M}_N$ with φ is an optimal strategy for $\mathcal{P}^n$, and vice versa, that is:*

$$\sigma_p^* = \arg\max_{\sigma_m}\{\Pr_{\mathcal{M}_N}^{\sigma_m}(\varphi)\},$$

where σ_p^* is given in Def. 4, and $\Pr^{\sigma_m}_{\mathcal{M}_N}(\varphi)$ is in Def. 7.

Proof. First, we observe that $\sigma_p = \sigma_m$, by construction. Second, one can derive, according to Definitions 4, 5, and 6 that $\cup_{\gamma_p} \Phi^{\sigma_p,\gamma_p,\mathcal{E}} = \cup_{\gamma_m} \Xi^{\sigma_m,\gamma_m}$ if $\sigma_p = \sigma_m$. The rest of the proof can be completed similarly to Theorem 1. □

Finally, we propose an algorithm to solve the problem of MDPSTs with LTL$_f$ objectives. This algorithm is based on a reduction to an MDPST with simple reachability, i.e., reaching a set of goal states. The algorithm is motivated by existing value iteration algorithms for MDPs with LTL$_f$ objectives [Wells *et al.*, 2020b]. Essential adaptations, however, are needed to handle the set-valued transitions in MDPSTs.

Given an MDPST $\mathcal{M}_N = (\mathcal{S}, s_0, A, \mathcal{F}, \mathcal{T}, \mathcal{L})$ and an LTL$_f$ formula φ, we first construct the corresponding DFA $Aut_\varphi = (2^{Prop}, \mathcal{Q}, q_0, \delta, acc)$ of φ. Then, the product MDPST $\mathcal{M}_N^\times = (\mathcal{S}^\times, s_0^\times, A^\times, \mathcal{F}^\times, \mathcal{T}^\times, \mathcal{L}^\times)$ is constructed accordingly with $\mathcal{S}^\times = \mathcal{S} \times Q, A^\times = A$. The set of goal states is given by $acc^\times = \{(s,q) \in \mathcal{S} \mid q \in acc\}$.

For the efficiency of strategy synthesis, we further introduce an optimization that computes an optimal robust strategy on a sub-MDPST, which only consists of states that are (forward) reachable from the initial state $s_0^\times$ and (backward) reachable from the set of accepting states $acc^\times$. To do so, we first partition $\mathcal{S}^\times$ wrt the initial state $s_0^\times$ and the set of accepting states $acc^\times$. Specifically, let $S_r \subseteq \mathcal{S}^\times$ be the set of states that can be reached from $s_0^\times$. We now partition $\mathcal{S}^\times$ into $\mathcal{S}^\times = S_n \cup S_d \cup S_p$, where $S_n = \mathcal{S}^\times \setminus S_r$ consists of states that cannot be reached from s_0, $S_d \subseteq S_r$ consists of states reachable from s_0 but that cannot reach any states in $acc^\times$, and $S_p = S_r \setminus S_d$ includes those that can be reached from both the initial and accepting states.

We construct a sub-MDPST $\mathcal{Z} = (\mathcal{S}_p, s_0, A_p, \mathcal{F}_p, \mathcal{T}_p, \mathcal{L}_p)$ from $\mathcal{M}_N^\times$ with respect to $\mathcal{S}_p$ as follows. $A_p = A^\times \cup \{a_\epsilon\}$, where a_ϵ denotes self-loop action. The set-valued transition function is such that $\mathcal{F}_p(s,a) = \mathcal{F}^\times(s,a), \forall s \in S_p \setminus acc^\times$ and $\mathcal{F}_p(s, a_\epsilon) = s, \forall s \in acc^\times$. The mass assignment function $\mathcal{T}_p$ is then given by i) $\mathcal{T}_p(s,a,\Theta) = \mathcal{T}^\times(s,a,\Theta), \forall \Theta \in \mathcal{F}_p(s,a)$ if $s \in S_p \setminus acc^\times, a \in A^\times$, and ii) $\mathcal{T}_p(s, a_\epsilon, s) = 1$ if $s \in acc^\times$.

Define a value function $V_\mathcal{Z} : \mathcal{S}_p \to \mathbb{R}_{\geq 0}$ by $V_\mathcal{Z}(s) = \max_{\sigma_m} \min_{\gamma_m} \{\Pr^{\sigma_m,\gamma_m}_\mathcal{Z}(acc^\times)\}$, which represents the maximal probability of reaching $acc^\times$ from s. Then one can get that $V_\mathcal{Z}(s) = 1, \forall s \in acc^\times$. For $s \in S_p \setminus acc^\times$, the Bellman principle of optimality is [Satia and Lave Jr, 1973]:

$$V_\mathcal{Z}(s) = \max_{a \in A_p(s)} \min_{\Pr(\cdot|s,a) \in \mathcal{H}_s^a} \Big\{ \sum_{s' \in \mathcal{S}_p} \Pr(s'|s,a) V_\mathcal{Z}(s') \Big\}. \tag{1}$$

Moreover, it was further shown in [Trevizan *et al.*, 2007] that a simplified Bellman equation exists for MDPSTs. That is, one can safely pull the min operator inside the summation, which gives a more efficient variant

$$V_\mathcal{Z}(s) = \max_{a \in A_p(s)} \Big\{ \sum_{\Theta \in \mathcal{F}_p(s,a)} \mathcal{T}_p(s,a,\Theta) \min_{s' \in \Theta} \{V_\mathcal{Z}(s')\} \Big\}. \tag{2}$$

An optimal robust strategy σ_m^* can be derived from $V_\mathcal{Z}$. Analogously to Sec. 3.3, due to cross product, every strategy σ_m for $\mathcal{M}_N^\times$ wrt $acc^\times$ induces a strategy for $\mathcal{P}^n = (\mathcal{N}, \varphi, \mathcal{E})$. Together with Thm. 3, we have the following.

Theorem 4. *Let $\mathcal{P}^n = (\mathcal{N}, \varphi, \mathcal{E})$ be as defined in Def. 4, $\mathcal{M}_N^\times = \mathcal{M}_N \times Aut_\varphi$ the constructed product MDPST, and $acc^\times$ the set of goal states. The computed optimal robust strategy σ_m^* of $\mathcal{M}_N^\times$ with reachability goal $acc^\times$ induces an optimal strategy σ_p^* for $\mathcal{P}^n$.*

Remark 1. *Let $\bar{\boldsymbol{F}} = \max_{s \in S_p}\{\max_{a \in A_p(s)} |\mathcal{F}_p(s,a)|\}$ be an upper bound of $|\mathcal{F}_p(s,a)|$ for all $(s,a) \in S_p \times A_p$. Let ϵ be the convergence precision of the value iteration (i.e., Eqn. (2)). The complexity for achieving an ϵ-suboptimal solution is $\mathcal{O}(|S_p|^2 |A_p| \bar{\boldsymbol{F}} \log \frac{1}{\epsilon})$. We highlight that MDPSTs are a subclass of MDPIPs, and they admit more efficient strategy synthesis algorithms than general MDPIPs. This is because for general MDPIPs, each round of value iteration of its corresponding Bellman equation (Eqn. (1)) involves two stages: 1) the inner minimization problem is solved (e.g., using a bisection algorithm) for each state and each action and 2) the value function is updated with dynamic programming.*

5 Implementation and Experimental Results

We implemented the solution technique described in Sec. 4, which subsumes the method described in Sect. 3, in Python, and use LYDIA [De Giacomo and Favorito, 2021] for LTL$_f$-to-DFA construction. The implementation details of our algorithms and experiments can be found on GitHub: https://github.com/piany/Tremblinghand_LTLf.

In this section, we present a case study to demonstrate the effectiveness of the proposed method. Our case study is based on the human-robot co-assembly problem, described in Example 2. The "trembling-hand" robot aims to stack blocks to a certain configuration with unpredictable human interventions. Note that we assume the human only has a limited number K of moves (otherwise, the robot has no way to guarantee task completion [He *et al.*, 2019].). In particular, we consider the configurations of having certain objects in certain locations. The goal configurations for N blocks are depicted in Figure 1, involving objects ranging from 2 to 6 (i.e., $|OBJ| \in \{2, 3, \cdots, 6\}$).

Implementation. The key challenge in the implementation is an effective representation of the planning domain with error actions. On one hand, it impacts the efficiency for model construction. As shown in [Wells *et al.*, 2021], the state space of the co-assembly problem grows exponentially in the number of objects. Therefore, it is extremely challenging to build a tractable model for a large number of objects. On the other hand, the efficiency of strategy synthesis is also impacted, as discussed in Remark 1.

In [Wells *et al.*, 2021], three different choices are examined to encode the state space, integer encoding (states are enumerated by breadth-first search), object encoding (use tuples, e.g., $(1, 2, 0)$, mapping each object to its location), and location encoding (using tuples, e.g., $(0, 1, 1, 0)$, mapping each location to the number of objects therein), where location encoding shows the best overall performance. We follow this observation and use location encoding for state-space representation. Note, however, that in our case a single tuple is

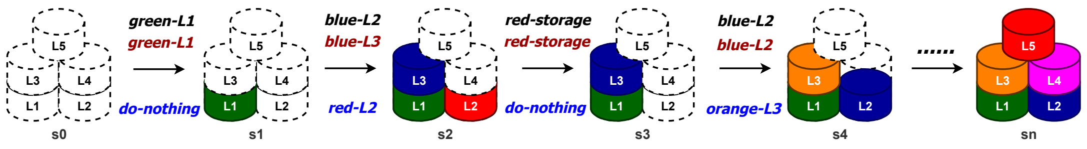

Figure 4: An execution example of an optimal strategy for the arch-building task. Robot-intended actions, robot-executed actions, and human interventions are shown in black, brick, and blue, respectively.

not enough to represent a state. This is due to the one-to-one correspondence between objects and locations in the goal configuration. For example, in the configuration involving 5 objects (where the goal configuration is shown on the upper right of Figure 1), if the locations of the green and blue blocks are interchanged, the task is deemed incomplete. To resolve this issue, we use a tuple of tuples for state encoding. For instance, a state $((0,0,0,1),(1,0,0,0),(0,1,0,0))$ means that Obj_1 is at location 3, Obj_2 is at the storage (the first element in each tuple represents storage), and Obj_3 is at location 1. Therefore, given N objects, the co-assembly domain has at most $2^{N(N+1)}$ states. Luckily, many of the states are invalid due to physical constraints, e.g., one block cannot be at two different locations, and each location (except for storage) can have at most one block. Using these physical constraints, we design a recursive algorithm (to avoid enumerating all possible states, hence greatly improving efficiency) to prune the co-assembly state space such that only valid states are maintained. The details of the state pruning algorithm can be found in Appendix A.

Moreover, the K-move limitation of the human should also be considered. We use a counter $\mathcal{C} = \{0, 1, \cdots, K\}$ to record the human moves. Together with the (pruned) state-space X of domain $\mathcal{N} = (\mathcal{S}, s_0, A, F_n, \mathcal{L})$, i.e., $X \subseteq \mathcal{S}$, we obtain an augmented state-space $\mathcal{S}' = X \times \mathcal{C}$. If the counter value is K, then the set of applicable human actions is restricted to DO-NOTHING for all $s \in \mathcal{S}'$. Finally, we can construct the probabilistic abstraction of the co-assembly problem in the form of an MDPST over $\mathcal{S}'$ following Sec. 4.3.

Figure 4 depicts a (simplified) execution example of an optimal strategy for an arch-building task with 5 blocks at 5 certain locations (see the right-most arch). All blocks are in storage (state $s0$) at the beginning. Then, the robot intends to put the green block at $L1$, which succeeds without any interference from the human (state $s1$). Next, the robot intends to finish building the base level by putting the blue block at $L2$. However, due to the trembling hand, the blue block was put to $L3$. Furthermore, the human put the red block at $L2$ to prevent the robot from building the arch (state $s2$). Note that the red block is supposed to be at $L5$ (see the right-most arch). In this case, the robot intends to remove the red block, which succeeds (state $s3$). After a finite number of executions, in spite of the trembling hand and the interventions from the human, the robot builds the arch (state sn).

Experimental results. In our experiments, the convergence precision for the value iteration in Eqn. (2) was set to 10^{-3}. All experiments were carried out on a Macbook Pro (2.6 GHz 6-Core Intel Core i7 and 16 GB of RAM).

We first show the effectiveness of the state-pruning technique. For the case that $K = 3$, the number of states and transitions in the constructed MDPST are shown in Figure 5 for different numbers of objects ($2 \leq |OBJ| \leq 6$). It is worth noting that six is the maximum number of objects that have been considered in the literature [Wells *et al.*, 2021] (due to the exponential blowup in the number of states). It is evident that the state space, post-pruning, exhibits much slower growth compared to exponential expansion.

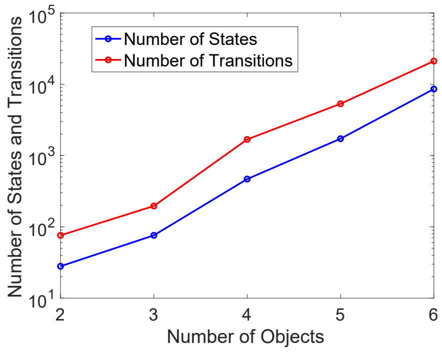

Figure 5: Number of states and transitions in the constructed MDPST for $2 \leq |OBJ| \leq 6$ (in log scale).

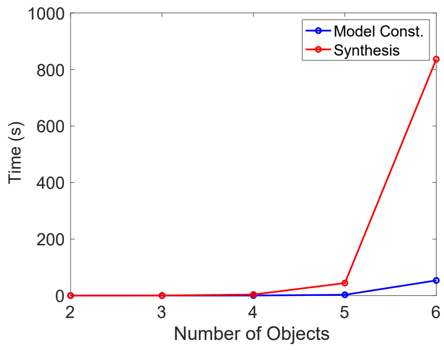

Figure 6: Computation time for model construction and strategy synthesis for $2 \leq |OBJ| \leq 6$.

In addition, the computation time for model construction (including state-space pruning and MDPST construction) and strategy synthesis (including DFA construction, product MDPST computation, and robust value iteration of Eqn. (2)) with respect to different number of objects ($2 \leq |OBJ| \leq 6$) are depicted in Figure 6. Note that LTL$_f$ synthesis in nondeterministic domains is 2EXPTIME-complete in the size of the LTL$_f$ formula and EXPTIME-complete in the size of the domain [De Giacomo *et al.*, 2023]. Moreover, the probabilistic behaviour of the agent (caused by the "trembling hand")

K	States	Transitions	Model Const. (s)	Synthesis (s)
3	1724	5324	3.222	45.81
4	2155	6655	4.685	78.820
5	2586	7986	6.958	132.42
6	3017	9317	9.283	190.203
7	3448	10648	12.383	263.826
8	3879	11979	15.768	347.015

Table 1: Model construction and synthesis computation times for 5 objects.

further complicates the synthesis problem. Nevertheless, one can see that, for the case of six objects, the overall problem (including model construction and strategy synthesis) can still be solved quite efficiently using the proposed algorithm.

For the case of 5 objects, we further explore situations where the upper bounds on human interventions range from 3 to 8 (i.e., $K \in \{3, 4, \cdots, 8\}$). The experimental results are shown in Table 1, where the size of the problem, i.e., the number of states and transitions, and the computation time for MDPST construction and strategy synthesis with respect to K, are listed. It is clear that, although both the number of states and transitions grow linearly as K increases (as does the model construction time), the synthesis time grows faster. This is because computing an ϵ-suboptimal solution for an MDPST $\mathcal{M}_N = (\mathcal{S}, s_0, A, \mathcal{F}, \mathcal{T}_N, \mathcal{L})$, which consists of reachable states only (following the partition optimization described in Section 4.3), is $\mathcal{O}(|\mathcal{S}|^2|A|\bar{\boldsymbol{F}} \log \frac{1}{\epsilon})$.

6 Conclusions

In this paper, we have investigated the trembling hand problem for LTL$_f$ planning in deterministic and nondeterministic domains. We formulate the problem formally by defining action-instruction errors and perturbed paths influenced by these errors. For the case of deterministic domains, we show that the problem can be reformulated as an MDP with an LTL$_f$ objective, leveraging existing algorithms for synthesis. In the case of nondeterministic domains, on the other hand, we utilise MDPSTs with LTL$_f$ objectives, for which we propose an efficient robust value iteration algorithm for synthesis. In particular, MDPSTs with LTL$_f$ objectives have been studied here for the first time. We also demonstrate the promising scalability of the proposed algorithm in a case study. For future work, we plan to leverage symbolic techniques for synthesis, aiming to improve efficiency.

Appendices

A State Pruning Algorithm

Given N objects, the co-assembly domain has at most $2^{N(N+1)}$ states. To obtain an effective representation of the co-assembly domain, we design a state pruning algorithm (Algorithm 1), which takes into account the physical constraints: (1) each block cannot be at more than one location, and (2) each location (except for storage) can have at most one block. In this way, only a set of valid states that satisfy these constraints are maintained for strategy synthesis.

Algorithm 1 *State Pruning*

```
Input: the number of objects $N$.
Return: the set of valid states $S_{valid}(N)$.
1: for $i = 2, \cdots, N$ do
2:     if $i = 2$ then
3:         $\mathbb{S} \leftarrow$ CreateStates$(i, i+1)$
4:         for $s \in \mathbb{S}$ do
5:             if RowsSumIsOne$(s)$ and ColsSumIsOne$(s)$ then
6:                 add $s$ into $S_{valid}(i)$
7:             end if
8:         end for
9:     end if
10:    if $i \geq 3$ then
11:        $r_{valid} \leftarrow$ CreateRow$(i)$
12:        for $s_{i-1} \in S_{valid}(i-1)$ do
13:            for $r(k)$ in $s_{i-1}$ do
14:                $r_{new}(k) \leftarrow \{r(k) + (0), (0, \cdots, 0, 1)\}$
15:            end for
16:            $S_{comb} \leftarrow (r_{new}(0), \cdots, r_{new}(i-2), r_{valid})$
17:            for $s_{expand}$ in Product$(S_{comb})$ do
18:                if RowsSumIsOne$(s_{expand})$ and ColsSumIsOne$(s_{expand})$ then
19:                    add $s_{expand}$ to $S_{valid}(i)$
20:                end if
21:            end for
22:        end for
23:    end if
24: end for
```

Note that Algorithm 1 employs a recursive approach, strategically avoiding the enumeration of all possible states. This recursive nature enhances efficiency.

Algorithm 1 takes as input the number of objects N and outputs the set of valid states $S_{valid}(N)$. It starts by computing $S_{valid}(2)$ for 2 objects (lines 2-9), followed by the recursive computation of $S_{valid}(\geq 3)$ through $S_{valid}(N)$ (lines 10-23). When $i = 2$, we first create all possible states $\mathbb{S}$ using *CreateStates*$(i, i+1)$, where each state is a 2×3 tuple and each element of the tuple is either 0 or 1 (line 3). Then for a state $s \in \mathbb{S}$, if all the rows sum equal to 1 (checked by *RowsSumIsOne*) and all the columns (except for column 0, which represents storage) sum equal to 1 (checked by *ColsSumIsOne*), then s is added to $S_{valid}(2)$ (lines 4-8).

When $i \geq 3$, we compute $S_{valid}(i)$ by expanding each state $s_{i-1} \in S_{valid}(i-1)$ (which is a $(i-1) \times i$ tuple) into a set of valid states s_i (which is a $i \times (i+1)$ tuple). In line 11, sub-algorithm *CreateRow(i)* returns a set of $1 \times (i+1)$ tuples whose row sum equals to 1 (i.e., r_{valid}). In lines 13-15, we expand each row $r(k)$ of s_{i-1} into a set of two rows, i.e., $r_{new}(k)$, by adding one element (either 0 or 1) to the end of $r(k)$. Then, in line 17, sub-algorithm *Product*$(\cdot)$ computes a set of tuples containing all possible combinations of elements from $r_{new}(0), \cdots, r_{new}(i-2)$ and r_{valid} (i.e., S_{comb}). For each s_{expand} in S_{comb}, if all the rows sum equal to 1 and all the columns sum (except for column 0) equal to 1, the state s_{expand} is added to $S_{valid}(i)$ (lines 17-21).

Acknowledgements

This work was supported in part by the ERC ADG FUN2MODEL (No. 834115), in part by the ERC ADG WhiteMech (No. 834228), in part by the EU ICT-48 2020 project TAILOR (No. 952215), in part by NSF grants IIS-1527668, CCF-1704883, IIS-1830549, CNS-2016656, DoD MURI grant N00014-20-1-2787, and in part by an award from the Maryland Procurement Office.

References

[Aineto *et al.*, 2023] Diego Aineto, Alessandro Gaudenzi, Alfonso Gerevini, Alberto Rovetta, Enrico Scala, and Ivan Serina. Action-failure resilient planning. In *ECAI*, pages 44–51, 2023.

[Altman, 1998] Eitan Altman. Constrained Markov decision processes with total cost criteria: Lagrangian approach and dual linear program. *Mathematical methods of operations research*, 48(3):387–417, 1998.

[Aminof *et al.*, 2021] Benjamin Aminof, Giuseppe De Giacomo, Alessio Lomuscio, Aniello Murano, and Sasha Rubin. Synthesizing best-effort strategies under multiple environment specifications. In *KR*, pages 42–51, 2021.

[Bacchus *et al.*, 1999] Fahiem Bacchus, Joseph Y Halpern, and Hector J Levesque. Reasoning about noisy sensors and effectors in the situation calculus. *Artificial Intelligence*, 111(1-2):171–208, 1999.

[Baier and Katoen, 2008] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[Bielefeld, 1988] R. Selten Bielefeld. *Reexamination of the Perfectness Concept for Equilibrium Points in Extensive Games*, pages 1–31. Dordrecht, 1988.

[Bonassi *et al.*, 2023] Luigi Bonassi, Giuseppe De Giacomo, Marco Favorito, Francesco Fuggitti, Alfonso Emilio Gerevini, and Enrico Scala. Planning for temporally extended goals in pure-past linear temporal logic. In *ICAPS*, pages 61–69, 2023.

[Brafman *et al.*, 2018] Ronen I. Brafman, Giuseppe De Giacomo, and Fabio Patrizi. LTL$_f$/LDL$_f$ non-markovian rewards. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *AAAI*, pages 1771–1778, 2018.

[Buffet *et al.*, 2005] Olivier Buffet, Douglas Aberdeen, et al. Robust planning with (l) rtdp. In *INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 19, page 1214. LAWRENCE ERLBAUM ASSOCIATES LTD, 2005.

[Cimatti *et al.*, 2003] Alessandro Cimatti, Marco Pistore, Marco Roveri, and Paolo Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *Artif. Intell.*, 147(1-2):35–84, 2003.

[Ciolek *et al.*, 2020] Daniel Alfredo Ciolek, Nicolás D'Ippolito, Alberto Pozanco, and Sebastian Sardiña. Multi-tier automated planning for adaptive behavior. In *ICAPS*, pages 66–74. AAAI Press, 2020.

[De Giacomo and Favorito, 2021] Giuseppe De Giacomo and Marco Favorito. Compositional approach to translate LTL$_f$/LDL$_f$ into deterministic finite automata. In *ICAPS*, pages 122–130, 2021.

[De Giacomo and Rubin, 2018] Giuseppe De Giacomo and Sasha Rubin. Automata-theoretic foundations of FOND planning for LTL$_f$ and LDL$_f$ goals. In *IJCAI*, pages 4729–4735, 2018.

[De Giacomo and Vardi, 2013] Giuseppe De Giacomo and Moshe Y. Vardi. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *IJCAI*, 2013.

[De Giacomo *et al.*, 2023] Giuseppe De Giacomo, Gianmarco Parretti, and Shufang Zhu. LTL$_f$ best-effort synthesis in nondeterministic planning domains. pages 533–540, 2023.

[Geffner and Bonet, 2013] Hector Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers, 2013.

[Ghallab *et al.*, 2004] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated planning - theory and practice*. 2004.

[Givan *et al.*, 2000] Robert Givan, Sonia Leach, and Thomas Dean. Bounded-parameter Markov decision processes. *Artificial Intelligence*, 122(1-2):71–109, 2000.

[Hahn *et al.*, 2019] Ernst Moritz Hahn, Vahid Hashemi, Holger Hermanns, Morteza Lahijanian, and Andrea Turrini. Interval Markov decision processes with multiple objectives: from robust strategies to Pareto curves. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 29(4):1–31, 2019.

[Haslum *et al.*, 2019] Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, and Christian Muise. *An Introduction to the Planning Domain Definition Language*. 2019.

[He *et al.*, 2019] Keliang He, Andrew M Wells, Lydia E Kavraki, and Moshe Y Vardi. Efficient symbolic reactive synthesis for finite-horizon tasks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8993–8999, 2019.

[Marchesi and Gatti, 2021] Alberto Marchesi and Nicola Gatti. Trembling-hand perfection and correlation in sequential games. In *AAAI*, pages 5566–5574, 2021.

[Nilim and El Ghaoui, 2004] Arnab Nilim and Laurent El Ghaoui. *Robust markov decision processes with uncertain transition matrices*. PhD thesis, University of California, Berkeley, 2004.

[Pnueli, 1977] Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977.

[Puterman, 2014] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. 2014.

[Reiter, 2001] Raymond Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT, 2001.

[Satia and Lave Jr, 1973] Jay K Satia and Roy E Lave Jr. Markovian decision processes with uncertain transition probabilities. *Operations Research*, 21(3):728–740, 1973.

[Trevizan *et al.*, 2007] Felipe W Trevizan, Fabio Gagliardi Cozman, and Leliane Nunes de Barros. Planning under risk and knightian uncertainty. In *IJCAI*, pages 2023–2028, 2007.

[Trevizan *et al.*, 2008] Felipe W Trevizan, Fábio G Cozman, and Leliane N De Barros. Mixed probabilistic and non-deterministic factored planning through markov decision processes with set-valued transitions. In *Workshop on A Reality Check for Planning and Scheduling Under Uncertainty at ICAPS*, page 62, 2008.

[Vardi, 2020] Moshe Y. Vardi. Efficiency vs. resilience: what COVID-19 teaches computing. *Commun. ACM*, 63(5):9, 2020.

[Wells *et al.*, 2020a] Andrew M. Wells, Morteza Lahijanian, Lydia E. Kavraki, and Moshe Y. Vardi. LTL$_f$ synthesis on probabilistic systems. In *GandALF*, volume 326 of *EPTCS*, pages 166–181, 2020.

[Wells *et al.*, 2020b] Andrew M. Wells, Morteza Lahijanian, Lydia E. Kavraki, and Moshe Y. Vardi. Ltl$_f$ synthesis on probabilistic systems. In *GandALF*, volume 326 of *EPTCS*, pages 166–181, 2020.

[Wells *et al.*, 2021] Andrew M. Wells, Zachary K. Kingston, Morteza Lahijanian, Lydia E. Kavraki, and Moshe Y. Vardi. Finite-horizon synthesis for probabilistic manipulation domains. In *ICRA*, pages 6336–6342, 2021.

[White III and Eldeib, 1994] Chelsea C White III and Hany K Eldeib. Markov decision processes with imprecise transition probabilities. *Operations Research*, 42(4):739–749, 1994.

[Zhu and De Giacomo, 2022] Shufang Zhu and Giuseppe De Giacomo. Synthesis of maximally permissive strategies for LTL$_f$ specifications. In *IJCAI*, pages 2783–2789, 2022.